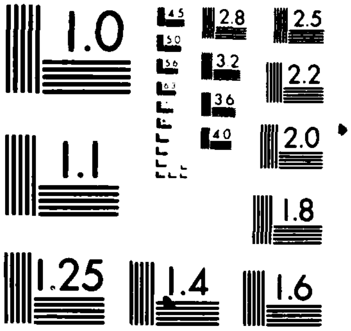
AD-A172 447 A SIMULATION ANALYSIS OF AN AUTOMATED IDENTIFICATION PROCESSOR FOR THE TA..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 1/3

UNCLASSIFIED R C MACFARLANE ET AL. JUN 86 F/G 17/7 NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

1

AFIT/GST/ENS/86M-13

A SIMULATION ANALYSIS OF AN AUTOMATED IDENTIFICATION PROCESSOR FOR THE TACTICAL AIR CONTROL SYSTEM

THESIS

Robert C. MacFarlane
Major, USAF

David M. McGuire
Captain, USAF

AFIT/GST/ENS/86M-13

Approved for public release; distribution unlimited

AFIT/GST/ENS/86M-13

A SIMULATION ANALYSIS OF AN AUTOMATED IDENTIFICATION PROCESSOR FOR THE TACTICAL AIR CONTROL SYSTEM

THESIS

Presentd to the Faculty of the School of Engineering
of tne Air Force Instititute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degreee of
Master of Science in Operations Research

Robert C. MacFarlane, B.A., M.B.A.
Major, USAF

David M. McGuire, B.S.
Captain, USAF

June 1986

Approved for public release; distribution unlimited

Acknowledgements

There are many individuals who deserve our thanks for their support during the months spent working on this project. First, thanks to our advisor and sponsor, Lt. Col. Joseph W. Coleman of the Air Force Human Resources Laboratory (AFHRL), for his guidance and timely suggestions. Thanks also to Maj. John R. Valusek for his inputs and suggestions as our reader from the Department of Operational Sciences at the Air Force Institute of Technology. Our gratitude also goes to Maj Raymond L. Spaeth and Rosemarie L. Preidis of the AFHRL for their sound advise as additional readers. Mr. Rick Bowman of the Command and Control section of the Foreign Technology Division aided us immeasurably in getting access to the TADZ computer code, in implementing it and in understanding its intricacies.

Finally and most importantly, we express our deep appreciation to our classmates, families, and friends for their support and encouragement. Special thanks to Debbie MacFarlane and Kathy Stewart (soon to be Mrs. McGuire) for their love and patience during the long hours required to complete this thesis.

Table of Contents

List of Figures

List of Tables

Abstract

Control of the airspace over the battlefield is a complex task. The control and reporting center (CRC), as part of the tactical air control system (TACS), plays a vital role in the air defense mission. The purpose of this thesis was to evaluate the utility of the proposed combat identification system - indirect subsytem (CIS-ISS), an automated identification feature, within the CRC. This issue was considered through a comparison of the automated system with the current manual system of identification. The primary measure of comparison was the number of hostile aircraft prosecuted during the first "wave" of a massive conventional attack.

Transient Air Defense Zone (TADZ), a large Fortran and SLAM based simulation model of the Soviet air defense system in use at FTD, was modified to represent the structure and operating procedure of the TACS. Parametric inputs were made to TADZ based on operational test and performance data for the CIS-ISS and the CRC. The model was then used to provide data for a statistical comparison between the automated and manual identification systems. The results showed that if the CIS-ISS is used in the envisioned centralized location with a "man in the loop," it will backlog under the load of a Central European threat. Distribution of the CIS-ISS from a centralized location and collection of more reliable input data are recommended areas for future effort.

I. INTRODUCTION

Background of the Study

Two primary missions of the United States Air Force are to "gain and maintain general air supremacy" and "to establish local air superiority" in the support of national policy decisions [DoD JSOG, 1984:1.14]. Since the conclusion of World War II, the U.S. has dedicated its political and military might to preserving the peace between the Warsaw Pact and NATO forces in Europe. In support of this aim, the U.S. has dedicated a significant amount of its military forces to the NATO alliance. Over the past forty years, advances in technology and changes in the military postures of the opposing forces have significantly altered the environment in Europe and thus have made the air defense mission of the NATO forces increasingly more complex in the European theater. It is estimated that in a direct conventional conflict between NATO and the Warsaw Pact more than 7,000 aircraft (2,000 NATO, 5,000 Warsaw Pact) could be in the air at any one time; and all this in an airspace comparable to the skies over the state of Oregon [Donnelly, 1985]! With the potential of having so many aircraft in this airspace, the task of providing adequate control of NATO's air forces is anything but a simple task. The Tactical Air Control System (TACS) was developed to aid in this mission.

Tactical Air Control System (TACS)

> The TACS is a network of facilities, equipment, people and procedures that permits the Air Force Component Commander to plan, direct and control tactical air operations. In short, it is a battle management system [Gardner, 1982:55].

This battle management system consists of surveillance radars, missile batteries, and control centers all joined together by communication links for the purpose of controlling the airspace over the battle area. With the great number of aircraft filling the skies during a conflict, the task of determining which are hostile and which are friendly is vital. With the advent of the computer to aid in the TACS process, this particular problem has become one of information gathering, filtering, and dissemination to the appropriate "decision nodes" of the air control system. With the aim of improving the timeliness and accuracy of aircraft identifications during wartime, the Air Force has contracted with the General Dynamics Corporation to develop the Combat Identification System - Indirect Subsystem (CIS-ISS) [A.F. Contract F19628-83-C-0054, 1983].

The CIS-ISS is an automated system made up of hardware and software designed to improve the aircraft identification function during wartime. The CIS-ISS will automate the data normally used to assist the operator in the "hostile-friendly" identification decision. It will also provide specific aircraft type identification with an associated probability of accuracy. CIS-ISS will be utilized in the control and reporting center (CRC) node of the TACS. A brief description of the process of the CRC, the senior radar control unit in the TACS, is provided below.

CRC System Process

> The CRC is the focal point for decentralized execution of air defense and airspace control functions. The CRC directs air defense operations, provides aircraft guidance or monitoring for offensive and defensive missions, relays mission changes to airborne aircraft and supervises the integrated activities of other radar elements [Gardner, 1982:55].

The CRC is the senior radar air defense node of the Air Force's tactical air control system. It provides radar surveiliance coverage for its assigned airspace. It is also tasked to identify and assign air defense resources against hostile and unknown aircraft and to ensure the safe passage of friendlies in and out of its area of responsibility. The CRC system process involves three basic activities that are performed in a sequence: surveillance, identification and weapons assignment.

Surveillance. [TACR 55-44, 1985:3.7-3.13] The first activity is the detection of a radar return when it appears on the radar scope. The scope surveillance operator (SSO) determines if the return is a valid one by observing it for one or two sweeps of the radar. If the return is valid, the SSO will initiate a radar track. The operator must continually update each track entered into the system. Tracks must be updated every 2 minutes or every 5 minutes depending on their priority. Priority 1 are hostile, priority 2 are unknowns, priority 3 are emergencies, priority 4 are air defense fighters, priority 5 are VIP flights, priority 6 are Special Missions and priority 7 are others [TACR 55-44, 1985:7.1]. As additional tracks enter the system, they are each initiated and updated in sequence.

Movements and Identification (M&I). [TACR 55-44, 1985:3.13-3.15] The second activity is track identification. Regulations state "the maximum time allowable for initial track identification is two minutes from the time the track is established by or received at the TACS element responsible for identification" [TACR 55-44, 1985:3.13]. M&I may be capable of identifying aircraft in less than two minutes if all available information is provided. The track must be identified as one of the following: assumed friend, assumed hostile or evaluated as unknown. M&I uses several pieces of data to correlate against the radar tracks for identification. They use flight plan data derived from the air tasking order (ATO) or frag orders, electronic identification of IFF/SIF (identification friend or foe / selective identification feature), airspace control procedures such as corridors and LLTR (low level transit routes), and visual identification (e.g., pilot visual inspection). Normally this information is available at the work station for the M&I operators to identify aircraft. Some data is provided via the console such as the IFF/SIF and the airspace control procedures. Other data must be handled manually such as the flight plan data. Here the operator must correlate speed, altitude and heading of the radar track with facts about aircraft scheduled to return to a specific recovery base at a specific time, altitude, heading, and corridor [TACR 55-44. 1985:3.13-3.14].

The M&I operator will receive all of the radar tracks pending identification. The operator will normally select the nearest radar track to identify first, the next closest, etc. After two minutes the operator must identify the radar tracks as assumed friend, assumed enemy, or evaluated unknown. The declaration of assumed enemy and evaluated unknown will initiate the third activity.

Weapons Assignment. [TACR 55-44, 1985:3.5-3.7] The third activity is interception. In the CRC, the weapons assignment officer (WAO) and the weapons controllers are the principal players in the interception activity. If a radar track is identified as hostile, the weapons assignment officer will assign a weapon controller to that track (this is assuming that the WAO has decided to use tactical air against the target and not air defense artillery) [TACR 55-

44, 1985:3.5]. The assigned weapons controller will then take steps to engage the target. For example, the controller may direct an interceptor from a combat air patrol to the target or an interceptor may be directed from an air base. The same basic procedure would apply for an evaluated unknown. A weapons controller would have to direct an aircraft from the ground to obtain a positive identification of the target. If the target is assigned to air defense artillery, a similar procedure would be followed. After all appropriate steps are taken to verify the hostile identification, the WAO will order the target destroyed by either an interceptor or a component of the air defense artillery.

Problem Statement

"The basic purpose of the CIS-ISS Demonstration Program is to validate the improved ID effectiveness which is predicted to result from the fusion of multiple sensors. ESM was chosen as one of the most potent sensors to be utilized for the demonstration [Schindall, undated:2]." The CIS-ISS will provide a more accurate identification (99% reliable) and allow the battle directors and weapon assignment officers to initiate tactical actions against hostile aircraft without having to visually identify the aircraft prior to ordering an engagement [Perini, 1985:81]. However, it is not known what the CIS-ISS response rate should be to ensure that aircraft identifications are being processed without backlog under medium and heavy workload conditions (e.g., the potential wartime environment).

Research Question

What is the range of the optimal response rate(s) that should be incorporated in the proposed CIS-ISS to ensure processing without backlog under combat conditions? The optimal response rate(s) identified will highlight the effects (trade-offs) with other system performance factors such as:

a. Track correlation accuracy (radar system versus CIS-ISS).

b. Manual versus automated interfaces.

c. Weapon system employment versus track identity (will more accurate track identity affect levels of air defense artillery or interceptor use).

Subsidiary Questions:

1. What are the limiting factors of the CIS-ISS to provide automated radar track correlation with available identification information (i.e. where is/are the bottleneck(s): the data links, information processing, operator proficiency, etc.)?

2. What is the arrival rate of aircraft into the TACS under combat conditions? This may require research into "exercise" results and possibly expert opinion in the case of insufficient data.

3. How many targets can the CRC operators "handle" in the manual mode prior to becoming saturated or overloaded? Again, this may involve expert opinion.

4. What are the roles and functions of each element of the TACS? This question will be answered with emphasis on the Surveillance and the Movements and Identification sections within the CRC.

Assumptions

Three major assumptions were made in developing the model for this thesis effort. First, it was assumed that the sensors that support and provide the input to the automated identification processor will be colocated with each radar site providing input into the CRC. This assumption will ensure that the identification sensors will match the surveillance coverage of the radars.

The second assumption is about the employment of the automated identification processor. The users of the CIS-ISS have indicated that they desire to retain a man in the loop and therefore the CRC will retain the identification function [Jones, 1985]. In addition, the system as it is now envisioned provides a single console for a CIS-ISS interface. Thus, a single central identification processor was modeled.

The third assumption concerned the environment in which the CIS-ISS model would be tested. The worst case scenario for the TACS system is assumed to be the NATO environment. Therefore, this initial research concentrates on a generic US

TACS operation within a central Europe type threat and uses the appropriate threat intensities, air traffic densities, etc..

Scope and Limitations

1. Data Gathering. The scope of the problem is limited by the data available. Data from the CIS-ISS feasibility demonstration at Eglin AFB in May 1985 is marginally suitable for use in the CRC model because of its raw form and the limited number of data points [Preidis, 1985]. When more expanded data on CRC service times becomes available from the CIS-ISS test conducted in Germany in the fall of 1985, the distributions will need to be recalculated. [Preidis, 1985]. In all cases, the data used was unclassified because one goal of this thesis was to keep the report unclassified if at all possible.

2. CIS-ISS Definition. The CIS-ISS/CRC interface is not yet finalized. Due to the ongoing research and validation of the CIS-ISS, the final specifications on this interface have not been agreed upon. The present situation makes the problem definition "broader" because of the lack of preciseness in the definition. The level of modeling as it pertains to the CIS-ISS will become more detailed as the interface is better defined by the agencies involved. The CIS-ISS SPO should be able to aid in defining the interface for the purposes of this study [O'Brien, 1985].

Overview of Remaining Chapters.

The remaining chapters parallel the research design employed in conducting this thesis. Chapter II is the literature review which covers both the analysis and the background investigation of the problem. Chapter III is a detailed description of the simulation methodology as applied to this particular problem. Chapter IV is a description of the computer model itself. In Chapter V, the results and statistical analysis involved in answering the research question will be discussed. Finally, in Chapter VI, conclusions are discussed based on the experimental results and recommendations are made regarding possible model refinements and future research.

II. LITERATURE REVIEW

Overview

This chapter presents a review of the literature that confirms the requirement for an automated aid for air defense identification. The literature search also establishes the use of simulation as a method of answering the research questions posed in Chapter One. Specifically the literature search was directed to answer the following questions:

1. What are the previous approaches to similar problems? A review of study efforts on the air defense system is presented. The applicability of this thesis effort is shown in relation to the overall research being conducted on the Tactical Air Control System.

2. Why is aircraft identification a problem in the modern air war? The military requirement for an improved capability to identify friend from foe is established. In addition, the background of the specific automated identification capability/electronic support measure that was tested at Eglin Air Force Base in Florida is presented.

3. Why use simulation? A general justification of applying the use of simulation as opposed to an alternative approach is answered. Included in the review is an analysis of the types of problems and the requirements for the use of simulation. A discussion of the advantages and limitations of simulation is also presented.

4. What is the appropriate "design of the experiment?" A literature search of alternative approaches to designing an experiment/simulation that offers the best design to produce data that will provide definitive or significant results. The discussion will include methods to reduce the number of experimental computer simulations required to produce a statistically valid output.

5. What are the appropriate statistical measures to be used? A review of the standard accepted mathematical and statistical tests are outlined with a brief description of the procedures applied.

Previous Research Efforts

This section will provide a brief overview on other research efforts involving similar topics for the TACS. Since this thesis effort is unclassified, the literature search addresses the material that is available in the public domain. The literature search was directed to locate available models of an air defense zone. The literature search revealed several models that can be classified into the following catagories.

Theater Models. There are several existing models that simulate the theater level war. These models simulate a ground and air battle for an entire geographical war zone and are usually aggregated at a high level. Such a high level model would include many air defense zone sectors. A single sector is the focus of this thesis effort. While some of the models can simulate the individual sector area of interest for this thesis, the models do not provide the means to analyze the position interaction required to answer the effect of the CIS-ISS on the TACS. The interaction between the various functional positions is not modeled to enough level of detail to be of use in this research effort. Interceptor War Game, Keen-Mixed Air Battle Simulator, and

STRAT DEFENDER are examples of the theater war models that model the air defense system at too high of a level to answer the research question of this thesis. Several of the current theater level models being used by the Department of Defense are listed in the Catalog of Wargaming and Military Simulation Models maintained by the Studies and Analysis Group at the Pentagon [Quattromani, 1982].

Single Conflict Models. Another grouping of models that simulate the combat results of the individual battles between single weapons systems are the single conflict models. An example would be the air battle between two aircraft (1 v 1) or two flights of aircraft (2 v 2/many). These models are usually used to optimize the weapons system capability of the individual weapons system. These models typically do not show the interaction of the multiple engagements or successive engagements that are required of the system model. Because of this limitation the models were not investigated further. Several of the specific system models are: Beyond Visual Range Air Combat, COLLIDE - An Aggregated Conversion Model for Air Combat, and Barrier Air Defense Model. These DoD models focus on system capabilities and do not treat the issue of varying the levels of identification capability [Quattromani, 1982].

Sector and Regional Air Defense Models. There are several models that simulate air defense systems of interest. The first model of interest is the QUEB model developed by

Alphatech Inc. for the US Air Force Systems Command, Foreign Technology Division under contract for the Command Control and Communications Systems Dynamics (C^3SD) project. "The purpose of the C^3SD project is to develop tools and techniques to assist scientific and technical (S&T) intelligence analysts in addressing C^3 problems [Merriman, 1983:i]." The other model of interest is the Transient Air Defense Zone (TADZ) model also developed by Alphatech. The major difference between the two models is that the QUEB model is an analytical model while TADZ is a simulation model. "QUEB is an analytical model used to calculate measures of effectiveness and performance for the C3 system, focusing on the time required to perform missions, the amount of penetrator leakage from the ADZ, and the level of resource utilization by the system [Merriman, 1983:10]."

A paper presented at the 6th MIT/ONR Workshop on C^3 Systems compared the two models [Merriman, 1983:18-22]. As noted in the report and in the introduction to the Software Report on TADZ, the TADZ model was developed to improve upon the draw backs of the QUEB model. The TADZ model allows the use of multiple types of interceptors and missiles to correct the QUEB deficiency which allowed only one type of penetrator and defense assets. In addition, the QUEB model assumed a steady state operational constraint when in reality the air defense cases needed to evaluate situations that did not fit a steady state model [Merriman, 1983:9].

Background Information. In addition to the research efforts mentioned above, several other efforts provide an excellent description of the environment and function of an air defense system. Two magazine articles provide general descriptions of the air defense system. The article "Employment of Tactical Air Control Radars" in the November 1982 issue of Signal Magazine by Colonel Robert E. Gardner provides an overview of the TACS. Colonel Gardner was Chief, Weapons Applications and Control Division, Directorate of Operations, Headquarters USAF when he wrote the article. A second article entitled "Tactical Air Control Simulator Correlates to Real World" in the April 1985 issue of National Defense also provides a discussion of the functions and interactions required to model an air defense system.

For the reader who is interested in learning the tactical terminology for the European theater, the paper "A Conceptual Design for Modeling the Air War in Central Europe" by Lt. Colonel Dennis L. Cole, provides a description of terms and a good framework for understanding the interactions of the army and air forces required to successfully prosecute operations across the war front [Cole, 1982].

The Problem of Identification

While the identification of specific deficiencies or requirements for an element of the TACS is not the intent of this thesis, the need for an improved capability to identify friend from foe is widely known. Headquarters Tactical Air Command has stated the requirements for an electronic support measures (ESM) sensor:

> the Tactical Air Control System (TACS) requires a survivable, passive sensor for the rapid identification of hostile and non-cooperative targets with a high degree of confidence. The Statement of Operational Need (SON), 305-79, Surface-to-Air, Combat Identification System-Indirect Sub-System (CIS-ISS) validates the need for this capability. The acquisition of an ESM sensor is an initial step to improve target identification of both cooperative and non-cooperative targets within the tactical C^2 structure [HQ TAC/DRC, 1985:1].

Dr. Joel Shindall described the need for an improved identification means in his paper entitled, "ESM Sensor for Non-cooperative Target Classification." Dr. Shindall first describes the capabilities of the radar system as follows.

> Modern radars excell in their ability to detect and track an airborne aluminum reflector, i.e. aircraft. however, since one reflection is pretty much like another, the radar by itself is ineffective at distinguishing the identity of a target, as depicted in Figure 1. This shortcoming is normally corrected by using an active IFF interrogator mounted on the radar to trigger coded responses from a transponder aboard each aircraft. The concern is that existing IFF techniques may be spoofed, jammed, or otherwise rendered ineffective in time of hostile engagement. The radar community has developed additional techniques which will undoubtedly be of some effectiveness, but it appears that conclusive target identification through radar alone is a risky proposition. Without reliable target identification, the most advanced weapon systems are rendered impotent [Shindall, undated:2].

The paper continues with a discussion of the benefits of an ESM capability and highlights the fact that the integration of the two systems would provide a highly efficient system.

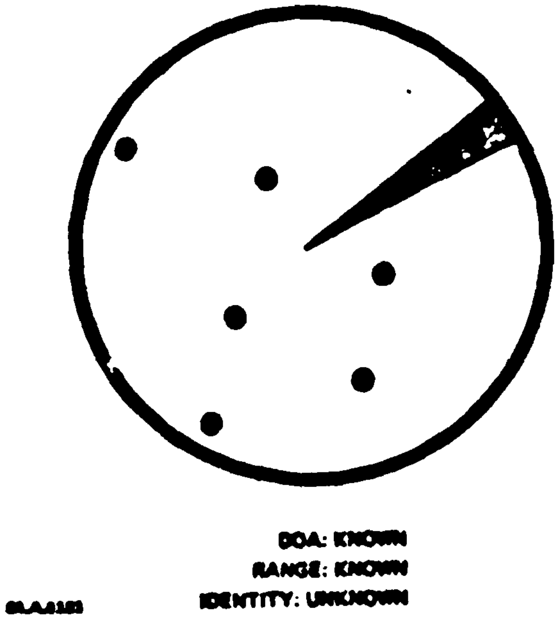


Figure 1. Radar Display (without IFF).

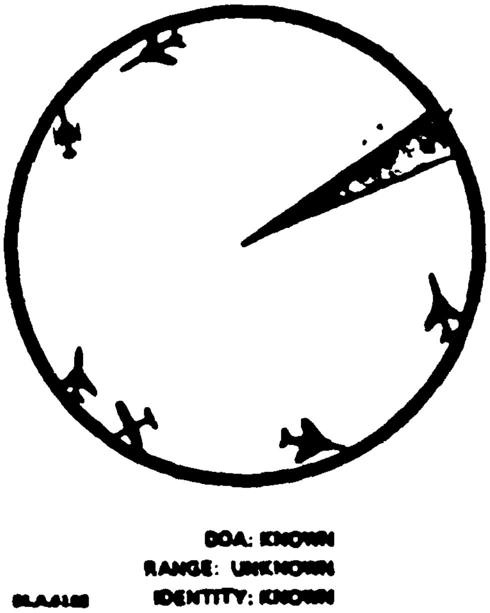


Figure 2. Passive ESM Display.

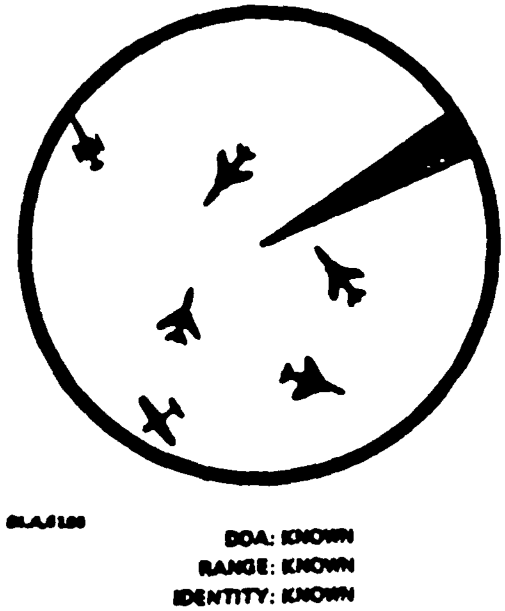


Figure 3. Radar Display Augmented by ESM Sensor.

> ESM techniques for aircraft identification utilize the fact that an aircraft, in order to do its job, is literally "glowing" in the electromagnetic spectrum. Unlike radar reflections which simply mirror the characteristics of the radar transmitter, the ESM emissions are highly characteristic of the particular on-board emitters carried by the aircraft. These characteristic emissions can be detected and identified by a properly designed and integrated passive ESM system, as depicted in Figure 2. The radar provides azimuth and range, while the ESM provides azimuth and ID. The fusion of these sensors provides all three, as diagrammed in Figure 3 [Shindal, undated:2].

Without the correct identification of all possible targets entering into the air defense system, the possibility of destroying our own forces is very likely. "Shooting up a few tanks and planes at today's prices could easily match the billions needed for a new IFF system. Until then, white flags should be the standard issue -- at least that standard is universally accepted [Defense Electronics, 1983:36].

The CIS-ISS was designed to provide a solution to the identification problem. Colonel Ewing, the Director of the Combat Identification System Program office at Wright Patterson Air Force Base Ohio, described the capabilities as follows: "Ninety-nine percent probability of correct ID is possible, using a combination of identification techniques [Perini, 1985:81]."

In an article in the Air Force Magazine entitled "Telling Ours from Theirs" [Perini, 1985], the point was made that no one system or techniques will satisfy all the operational requirements. The article makes a point of showing that the organization of the United States Combat Identification System is composed of many diverse elements as the Table I below indicates.

Table I.

Organization of the U.S.Combat Identification System

	CIS	
Direct Subsystem	Indirect Subsystem	Non-cooperative
--Mark X	--JTIDS	--Data obtained on unidentified aircraft using special sensors and processing techniques
--Mode S	--E-3A (AWACS)	
--Mark XII	--TPS-43 Radars	
--Mark XV	--ESM	
--Other onboard sensors	--Other friendly sources of data	

Why Simulation

Justifying the selection for the use of simulation over some of the more classical methods of operations research is the purpose of this section. Before proceeding further, the following definitions from the glossary of *Venture Simulation in War, Business, and Politics* are presented.

> *Simulation* 1. An operating representation of events and processes.
>
> 2. A technique used to study and analyze the operation and behavior, by means of models of systems conditioned by human decision and/or probabilistic natural influences [Hausrath, 1971:318].

Model 1. A representation of a real situation in which only those properties believed to be relevant to the problem being studied are represented.
2. A representation of an object or structure; and explanation or description of a system, a process, or series of related events [Hausrath, 1971:315].

Only one other term needs to be clearly defined at the outset, and that is gaming. The difference between gaming and simulation is the active participation of human beings during the exercising of the gaming model [Greenberg, 1981:95]. In the recently published text The Military Applications of Modeling, the difference between wargaming and simulation is explained:

> "Wargaming" precisely applies only to the analysis of combat situations in which human players form a part of the decision process. Hence, as defined in this text, models are used to support wargame analysis, but are not wargames themselves. "Simulation" is a type of model in which the objective is generally to replicate a reasonably well understood process, and for which uncertainties are treated by Monte Carlo methods. It hence assumes sufficient knowledge about the process to at least specify its dynamics and the form of its probability distribution [Battilega and Grange, 1984:14].

Now that the terms to be used are defined, the focus will be on why simulation is used as the basic methodology in this thesis. First, the complexity and cost of conducting a real experiment to test the research question is prohibitive. The second reason for using simulation is the ability to explore excursions or variations. The third reason is that the system under study is well defined and understood. The remainder of this section shows that these ideas are not just held by the authors but are widely accepted and practiced in the operations research community.

Complexity Requires Simulation

Norman C. Dalkey wrote as an introduction to his chapter on simulation in *Systems Analysis and Policy Planning: Applications in Defense*

> Simulation is a technique for studying complex military processes. It consists of an abstract representation of the more important features of the situation to be studied, designed to be played through in time either by hand or by computer [Quade and Boucher, 1977:241].

In some cases it is not so much the complex nature of the problem that requires simulation, rather social or economic cost prohibits the testing or experimentation required to produce a data base upon which to base a decision.

> War is an example of the complexity, the uncontrolled variability, and the impossibility of obtaining and recording desired data through manipulation and observation. Models make it possible to examine, manipulate, and analyze certain aspects of performance with greater precision and ease than is permittted by observation of the real-life process. A model, serving as a substitute for and a simplification of the real-life process, brings the task to manageable dimensions [Hausrath, 1971:98].

The very process which models sometimes represent are not fully explorable short of war or economically unacceptable experiments [Battilega and Grange, 1984:8]. Since the social cost and complexity of designing a war just to evaluate this system is not a feasible solution, the simulation alternative was selected.

Simulation as an Experiment

Choosing simulation as a method of studying a problem can not be based only on the fact that the complexity of a live experiment is infeasible. Even if the right situation could be found to test the system, a valid experimental test requires the capability to reproduce the results under the same circumstances and the need to test the experiment under a range of conditions.

> The fact that the assumptions of the model are explicit and the results can be duplicated is extremely important when a sizable community with differing interests (for example, the various agencies of the Department of Defense) is interacting on a problem [Quade and Boucher, 1977:250].

Frequently the answer to a problem may require the analysts to evaluate the situation in alternative modes. "For example, there may be a need to analyze performance under various terrain conditions; or alternatively, a senstitivity [sic] analysis involving many modifications of equipment may be called for. [Shubik, 1975:281]" In fact, sensitivity analysis is a primary reason for choosing simulation as a methodology. If the model and the simulation is conducted properly the results can be shown to be valid over some range of input values. This analysis of the sensitivity of the model is required to show that the results obtained reflect not just one limited situation but remain valid over some known and defined range of values based on the assumptions of the initial problem.

> Ordinarily there is no unique, "best" set of assumptions, but a variety of possibilities, each of which has some basis for support. A good systems study will include sensitivity test on the assumptions in order to find out which ones really affect the outcome and to what extent. This enables the analyst to determine where further investigation of assumptions is needed and to call the attention of the decision maker to possible dangers that might be present [Quade and Boucher, 1977:423].

Knowledge of the System Required

Martin Shubik describes the requirements to model a system in his book Games for Society, Business and War, Toward a Theory of Gaming. In talking about the use of operations research and tactical simulation, Shubik states "The requirements of a simulation of this variety is that the system to be simulated is relatively well-defined and that its components can be accurately described and mathematically modeled [Shubik, 1975:12]."

The need for the system to be modeled to be fully understood is also reflected in the description of the anatomy of a model in Venture Simulation in War, Business and Politics.

> A model, serving as a substitute for and a simplification of the real-life process, brings the task to manageable dimension. The model builder must, however, understand the entire process and the relation of the component systems. The model builder (one of the gaming specialists) builds models piece by piece as components of a larger, complex system [Hausrath, 1971:98].

The same requirement is also stated in Introduction to Simulation and SLAM II. When describing the simulation process, Pritsker says that the building of the model requires that the system to be modeled be abstracted "into mathematical-logical relationships in accordance with the problem formulation [Pritsker, 1984:10]." "The modeler must understand the structure and operating rules of the system and be able to extract the essence of the system without including unnecessary detail [Pritsker, 1984:11]." The authors will now show that the air defense systems and functions are well understood and defined.

The successes in acquiring complex and expensive weapons systems for air defense have been attributed to the well defined functions within the air defense mission area. In the article "C^3I Evolution Leaves a Lot to Chance" in Defense Electronics, air defense systems are described as a text book example of successful acquisition because of the well defined functions and boundaries between the component systems.

> Operational missions are narrowly stated and well understood: target detection, tracking and identification; threat evaluation; weapons assignment and control; and engagement assessment [Ablett, 1984:49].

The article continues by stating that just this type of explicitly defined systems functions are required to do successful system engineering. Effective systems design requires such a well understood and well documented description in order to decide which sensors will be required for specific users.

> For automating the air defense functions, the man-machine interface was relatively easily achieved. Positional information could be conveniently and naturally presented on a plan-position-indicator (PPI) scope, which looks like a map, an analog to the "real" world. Symbols could be used to represent the various catagories of objects of concern. Amplifying data and choices were also limited, and could be identified in advance [Ablett, 1984:54].

All of these factors led to the development of efficient work stations for the functional positions in the air defense mission.

Experimental Design

> Experiments are carried out by investigators in all fields of study either to discover something about a particular process or to compare the effect of several factors on some phenomena. In the engineering and scientific research environment, an experiment is usually a test (or trial) or series of tests. The objective of the experiment may either be confirmation (verify knowledge about the system) or exploration (study the effect of new conditions on the system) [Montgomery 1984, p 1].

The above quote is the opening to the Montgomery's book Design and Analysis of Experiments and provides a good overview of what an experiment should be. The purpose of this thesis experiment is to explore the effect of the new condition (automated identification) on the system (the existing TACS). The purpose of experimental design, like the scientific method, is to provide a logical and systematic approach to problem solving. Although the majority of the experimental design specifics will be discussed in Chapter V, this section will briefly discuss the benefits of experimental design.

> If an experiment is to be performed most efficiently, then a scientific approach to planning the experiment must be employed. By the statistical design of experiments, we refer to the process of planning the experiment so that appropriate data will be collected, which may be analyzed by statistical methods resulting in valid and objective conclusions. The statistical approach to experimental design is necessary if we wish to draw meaningful conclusions from the data [Montgomery, 1984:2].

Another advantage of experimental design is the ability to analyze multiple factors or influences on the system with a minimum number of observations.

> We want to return briefly to the advantages of a factorial design as compared with the one-factor-at-a-time method. Suppose we take N observations. To measure the main effect of the first factor we take N/2 observations at its low level and N/2 observations at its high level. In a factorial design we distribute the N/2 observations at the low level of factor 1 evenly between the high and low levels of the remaining (k-1) factors; the same for the N/2 observations at the high level of factor1. The precision of our estimate of the main effect of the first factor will not change, but the factorial design makes it possible to estimate the effects of all the other factors at the same time. [Kleijnen, 1975:319]

Statistical Measures

Comparing Alternative Systems. This section will outline the statistical methods that detect the differences between the two populations. The actual equations used will be explained in further detail in Chapter V. In operational terms, a difference between populations means that there will be some measurable difference between the outputs of the model with and without the automated identification feature. The difference will not only be detected but also ranked to show one system's measure of performance is higher or lower than the other. In statistical terms this is called ordering populations.

> Users of statistical methods as well as teachers and students of statistics need to become acquainted with ranking and selection procedures, since these techniques answer a question that is raised in many investigations but seldom answered by the more traditional methods of analysis. In layman's terms, the question might be one of the following: Which one (or ones) of several well-defined groups is best (in some well-defined sense of best)? Which of several alternative courses of action is best? [Gibbons et al, 1977:vii]

Law and Kelton in Simulation Modeling and Analysis recommend the use of confidence intervals between the two measures of merit produced by a simulation experiment involving different systems. They recommend the use of a confidence interval over a difference of means test. A difference in means test

> results only in a 'reject' or 'fail to reject' conclusion, a confidence interval gives us this information (according as the confidence interval misses or contains zero, respectively) as well as a measure of the degree to which the system responses are likely to differ, if at all. [Law and Kelton, 1982:319]

Law and Kelton also point out that the general procedure in selecting the best of k systems involves building a confidence interval around the probability of making the 'correct selection' about which system is best. The procedure developed by Dudewicz and Dalal

> involves 'two stage' sampling from each of the k systems. In the first stage we make a fixed number of replications of each system, then we use the resulting variance estimates to determine how many additional replications from each system are necessary in a second stage of sampling in order to reach a decision. [Law and Kelton, 1982:322]

As will be shown, the problem of selecting the correct number of runs or samples to use is dependent on the procedure to be employed and the purpose of the analysis. The reason the sample size is not explicitly stated is the value selected determines the reliability of the experiment and is a choice of the person conducting the experiment and the statistical procedure used.

Kleijnen, in Statistical Techniques in Simulation, divides his survey of statistical procedures to be used in analyzing simulation data into three catagories. The first catagory is calculating the reliability of a signal population given a number of samples (simulation runs). The second catagory is multiple comparisons procedures when the sample sizes or the data is given and the analyst wants to select or order the 'k' populations. The final catagory is multiple ranking procedures where the objective is to determine the number of samples necessary to measure a selected difference with a given probability of being correct. [Kleijnen, 1975:451]

The reader is cautioned at this point to make the distinction between the general categorization of procedures described in the previous paragraph. The first category of procedures is concerned with "problem of determining the reliability of statements on the mean of one population, or the difference between the means of two populations, the sample size being fixed [Kleijnen, 1975:525]." Note that when determining the difference between the means we are not concerned with the actual value of the individual means, only that some measurable difference exists.

In the second category, multiple comparisons, the general use of these procedures is to compare multiple factors in the initial screening of an experimental design. The experimentor has an initial set of data, 'k' populations with n_i (i = 1,2,...,k) samples in each population and the experimentor wishes to determine which populations have values better than some level. This type of screening is applicable when the experimentor is not yet looking for the best system, rather the goal is to find which factors/levels influence the system output. [Kleijnen, 1975:525]

The final category of procedures, multiple ranking, involves the selection of the 'best population'. The techniques determine the number of observations necessary to make the 'best selection'. Implicit in these procedures is the idea of an 'indifference zone', where if the difference is not large enough to be outside of this zone, the experimentor does not want to spend the effort to take required number of observations to detect that small difference, rather he is willing to accept with a high probability that the populations do not differ. In other words, he is indifferent . [Kleijnen, 1975:599-601]

Kleijnen also further divides the multiple ranking procedures based on the types of assumptions made: independent observations, approximation of normal distributions, and the type of variance (common, unequal or unknown) [Kleijnen, 1975:604-605]

Gibbons, Olkin, and Sobel also describe many procedures in Selecting and Ordering Populations, A new Statistical Methodology. Gibbons, Olkin, and Sobel describe the

> analytical aspect of the problem of selecting the best population. These problems are called the determination of sample size, the calculation of the operating characteristic curve, and the estimation of the true probability of a correct selection. [Gibbons et al, 1977:18]

In general, they recommend the use of operating characteristic curves that give a sample size based on the number of systems compared, the delta (difference between the means) to detect and the probability of making the correct choice [Gibbon et al, 1977:18-19]. It is important to note that Gibbons, Olkin and Sobel have assumed that the first stage of sampling has already been accomplished, i.e. they are given the 'k' populations with $n_{[i]}$ samples.

Summary

This chapter has presented a review of the appropriate literature to both confirm the requirement for an automated identification system and to establish the use of simulation as one satisfactory method of answering the research question posed in Chapter One. In addition, a review of the literature covering experimental design and statistical techniques was presented. Chapter Three will explain the methodology used and expand on the methods found in the literature review.

III. METHODOLOGY

Overview

In this chapter the specific steps taken in the research will be discussed and documented. As discussed in the literature review, computer simulation was chosen as the main technique to approach this problem. Of the several reasons for using simulation to solve this problem, perhaps the most important one is sighted by Banks and Carson in their text, Discrete Event System Simulation: "simulation can be used to experiment with new designs or policies prior to implementation, so as to prepare for what may happen [Banks and Carson, 1984:4]."

Under the sponsorship of the Air Force Human Resources Laboratory (AFHRL) at Wright Patterson AFB, OH, this research effort was attempted to answer the question of whether or not the proposed CIS-ISS will "improve" the identification function in the TACS. The advantages of modeling the interaction effects prior to purchasing this system should be clear. For example, if the new system can be shown not to improve the identification function, the unnecessary purchase of a costly system could be avoided. The simulation model developed in this effort will also be used by the AFHRL to help answer human interaction questions. So, although answering the research question at hand is the main purpose of this thesis, some long term benefits should also be realized.

This chapter will cover the research design of the thesis effort. The background study of the problem, the model construction and verification, and the experimental design will all be described as they relate to this thesis effort. Within the section on experimental design, the statistical techniques used will be covered in detail. Finally, there will be a short discussion on some of the techniques that helped keep the thesis work "on track"!

The Research Design

The research plan or design of this thesis effort is not unique. In fact, the simulation process outlined by A. Alan B. Pritsker is very much representative of the process this effort follows. Table II shows the ten steps cited by Pritsker.

Table II.

The Simulation Process [Pritsker, 1984:10-13].

1. Problem Formulation
2. Model Building
3. Data Acquisition
4. Model Translation
5. Verification
6. Validation
7. Strategic and Tactical Planning
8. Experimentation
9. Analysis of Results
10. Implementation and Documentation

These steps outlined by Pritsker are not the only approach to a simulation thesis. Shannon provides a similar list in his article, "Simulation: A Survey with Research Suggestions" [Shannon, 1975:289-290]. Banks and Carson also discusses the steps in the simulation process in their text [Banks and Carson, 1984:11-14]. This analysis will follow a process which includes the major elements found in all of these suggested outlines.

Problem Formulation/Background Study. The actual problem identification in this study was originally posed by the AFHRL. The question of what effect the CIS-ISS would have on the TACS was a logical one from the start. Although significant research was done regarding the methodology to be used, the simulation approach was a major consideration from the beginning owing to the fact that the AFHRL preferred to have a well constructed model of the CRC at the conclusion of the thesis effort.

The relevance of this problem is documented in Chapter II. The background study of the problem served several purposes. One purpose was to familiarise the authors with the TACS operation from the macro level all the way down to the specifications on the communications links. This knowledge has proven especially valuable in the construction phase of the model. Another benefit of the background study was to gain insights from past studies in the same subject area (the TACS).

Data Collection. The main thrust of the data collection has been in the area of operational tests of the CIS-ISS. The specific data elements required pertain mainly to the arrival rates of hostile aircraft for a NATO type scenario and to the reaction/service times of the various operators within a control and reporting center or CRC. The first data available came from an operational test of the CIS-ISS at Eglin AFB, FL, in May of 1985 [Preidis, 1985]. Due to the limited number of observations in some cases, this data will need to be checked against other operational test data as it becomes available. The CIS-ISS tests conducted in Germany during the fall of 1985 may provide "good" data in that the AFHRL specified some of the data elements to be collected [Preidis,1985].

Having a source of data is only the beginning of the data collection process. Arranging or presenting this raw information in a useful form is the next step. There are many ways to present data for use in a computer simulation. For example, the time required to identify a radar track could be represented as a constant average, a table of probable values, or a parametric distribution [Innis and Rexstad, 1983:7]. Each situation will dictate which is the more efficient method based on factors such as the underlying population, model sensitivity to that parameter, and data availablity. These particular issues are explored further in Innis and Rextad's article on model simplification [Innis and Rextad, 1983:7-10]. As a starting point, all the data was modeled as parametric distributions. The sensitivity of the model to the inputs determined their final form.

Input data analysis is a subject addressed in many statistical and simulation texts. The following references were used in this effort: Banks and Carson, 1984; Hines and Montgomery, 1980. The basic process in identifying/fitting a distribution to data is described in the Banks and Carson text and is reproduced here in Table III.

Table III.

Input Data Analysis [Banks and Carson, 1984].

1. Identify the distribution
    - histograms
    - distributional assumption
2. Parameter estimation
    - sample mean and variance
    - suggested estimators
3. Goodness-of-fit tests
    - chi-square test
    - Kolomogorov-Smirnov (K-S) test

The above listed process can be done easily either by hand, albeit time consuming, or with the help of statistical packages found on most computer systems. For this effort the SAS package (available on the VAX/11-785 VMS system at AFIT) was used [SAS Applications Guide, 1980].

The goodness of fit tests previously described have validity only under certain conditions. Specifically, the chi-square test is valid for larger sample sizes, for discrete and continuous distributional assumptions, and when maximum likelihood estimators are used [Banks and Carson, 1984:350]. The K-S test is "particularly useful when sample sizes are small and when no parameters are estimated by the data [Banks and Carson, 1984:357]." Most of the data requiring distributional description in this effort fit the assumptions required for the chi-square test (i.e. the continuous distributional assumption and relatively large sample size). It is also generally agreed that a minimum expected frequency of 3, 4, or 5 be used for the class intervals in a chi-square test [Banks and Carson, 1984:350; Hines and Montgomery, 1980:299]. Although a strict definition of a "large" sample size does not exist, Banks and Carson suggest that a for sample of less than 20, the chi-square test should not be used [Banks and Carson, 1984:351]. This being the case, the chi-square test was used almost exclusively in the input data analysis of this project. The final distributional forms of the input parameters can be found by referring to the computer code in Appendix A.

Model Construction. The actual building of the computer model proceeded through several different phases. The early stages of model building were based on a simplified conceptualization of the CRC within the TACS. The actual coding was done entirely using the SLAM 'network' approach, with no external FORTRAN coding required. The entities were defined as radar tracks flowing through the network from the track initiation event to the eventual destruction of the track if identified as hostile. The arrival rates of the tracks and the service times in the system were based on the data received from the May '85 operational tests of the CIS-ISS at Eglin AFB. This initial model did not contain the level of detail required in several areas. First, the data used for the service time distributions was not considered adequate because of the limited number of observations. The modeling of the fighter and surface to air missile engagements was also limited to a very low level of detail. Although this first cut at the model provided valuable insight into the inner workings of the Control and Reporting Center, the authors felt that more detail was required in the previously mentioned areas to instill more confidence in the model output.

Through the process of research and literature review, an article by Merriman and Dowdle of ALPHATECH, Inc. provided a comparison of two air defense command and control models [Merriman and Dowdle, 1982]. After several phone calls to ALPHATECH, it was learned that the models were being used by the Command and Control section of the Foreign Technology Division (FTD/TQC) at Wright Patterson AFB. FTD/TQC is using these two models in their work on the Soviet Air Defense system. The two models compared in the article were Queuing Based (QUEB) and Transient Air Defense Zone (TADZ). QUEB is an analytical queuing theory model of a RED air defense zone. TADZ is a SLAM, FORTRAN, and Pascal based computer model of the RED air defense zone and its response to a BLUE force attack. Through an examination of the documentation and computer code of TADZ, it was determined that several portions of the model could be adapted to model the U.S. TACS. TADZ provides the level of detail required that was missing in the first attempt at modeling the CRC.

The TADZ model runs on a VAX/11-780 with a VMS operating system at FTD and is completely compatible with the computer and operating system available at AFIT. Any requests for access to the TADZ coding or documentation should be referred to FTD/TQC. A masters thesis effort by Larson and Vane at the Naval Postgraduate School was very helpful in providing an understanding of the TADZ model [Larson and Vane, 1985].

The experience gained from the first attempt at modeling the system paved the way for successful incorporation of the appropriate the CRC into TADZ. Throughout the model building process care was taken not to make the common errors that can beset any model builder. Innis and Rexstad's article on model simplification techniques was very helpful in pointing out some of the possible dangers and in providing guidance for keeping the model at the appropriate level of detail [Innis and Rextad, 1983]. Coding improvements, logic analysis, and variance reduction are just a few of the efficiency hints discussed in this article that were helpful.

Appendix A contains the actual code of the final CRC model. For a detailed description of the CRC model refer to Chapter IV of this thesis.

Experimental Design. From the authors' point of view, one of the important reasons for planning or designing the approach to be taken in this thesis is economy of effort. Hicks states that in the design of an experiment, "one seeks to obtain the maximum amount of reliable information at the minimum cost to the experimenter [Hicks, 1973:2]. Montgomery defines the statistical design of experiments as:

> the process of planning the experiment so that appropriate data will be collected, which may be analyzed by statistical methods resulting in valid and objective conclusions. The statistical approach to experimental design is necessary if we wish to draw meaningful conclusions from the data. [Montgomery, 1984:2]

Montgomery also recommends that everyone involved in the "experiment" have a clear vision in advance of what is to be researched, the type of data to be collected, and a general idea of how the data will be analyzed [Montgomery, 1984:3]. In Hick's text, Fundamental Concepts in the Design of Experiments, the author suggests the outline shown in Table IV as one possible approach to experimental planning [Hicks, 1973:4,5].

Any textbook dealing with the subject of experimental design will have essentially the same elements included in Table IV by Hicks. It was decided to follow this particular process in the design of the experiment. In reality, the order of accomplishment may have varied slightly during the process, but the intent was to follow the general guidelines.

Table IV.

Experimental Planning.

I. Experiment

A. Statement of problem

B. Choice of response or dependent variable

C. Selection of factors to be varied

D. Choice of levels of these factors

1. Quantitative or qualitative

2. Fixed or random

E. How factor levels are to be combined

II. Design

A. Number of observations to be taken

B. Order of experimentation

C. Method of randomization to be used

D. Mathematical model to describe the experiment

III. Analysis

A. Data collection and processing

B. Computation of test statistics

C. Interpretation of results for the experimenter

In the choosing of a *response variable*, the early work on the basic CRC model was very valuable. The number of hostile radar tracks that "get through" the air defense net was the first response variable identified. The second response variable chosen was the number of incorrect identifications by the "system." Keep in mind that the question to be answered is whether the CIS-ISS will provide a significant improvement in the identification process. These two response variables will be the indicators of the improvement or lack of it.

The process of selecting the factors to be varied was continuous in nature. During the model verification phase the model's sensitivity to the input parameters was tested using statistical comparisons on the mean response [Banks and Carson, 1984:Ch 10]. An example of a factor that is varied in this analysis is the arrival rate of aircraft/tracks into the CRC's coverage area. The decisions on exactly how to vary the different factors in the model were made based on experience and from data gathered in the background study pertaining to system performance, resource levels, etc..

Determining the number of observations to be taken is a decision that can be aided by statistical analysis. The choice of sample size is a topic covered widely in the literature. Montgomery, Banks and Carson, and Hicks all provide acceptable methods for determining sample size for one, two, and several factor designs [Montgomery, 1984; Banks and Carson, 1984; Hicks, 1973]. Fishman provides some useful insights on the subject in his Management Science article on estimating sample size [Fishman, 1971:21-38]. The general technique involves the use of Operating Characteristic Curves found in most statistical analysis and experimental design texts.

As previously discussed, the thrust of the final analysis is to compare the system performance with and without the CIS-ISS. The plan for making the experimentation "runs" reflects this fact. Identical runs are made with and without the computer aided identification feature and the results (the response variables) are tested for statistical difference or variance. This leads the discussion into the area of factorial design and the use of the ANOVA (analysis of variance).

Again, the literature contains several texts and articles dealing with ANOVA and statistical analysis of the output from a simulation model. The following authors address the subject: Law and Kelton, 1982; Hicks, 1973; Banks and Carson, 1984; and Montgomery, 1984]. Carson and Law have written an excellent article in the Operations Research journal on the subject of confidence intervals using the 't' statistic and the use of blocking in factorial design [Carson and Law, 1979:1011-1025].

Another topic of interest in experimental design is variance reduction. The various "techniques or tricks" in this area are meant to improve the statistical efficiency of the experiment [Banks and Carson, 1984:487]. Common random numbers and antithetic random numbers are two of the more widely discussed methods. Kleijnen discusses these two techniques in his Management Science article [Kleijnen, 1975:1176-1185]. Law and Kelton devote an entire chapter in their text, Simulation

Modeling and Analysis [Law and Kelton, 1982:Ch 11]. Due to the complexity of the CRC model, a combination of both the antithetic and common random number techniques was used in this experiment.

Verification and Validation. As noted earlier, it is important to have a vision of the experimental design prior to performing the experiment itself. A great deal of the previous discussion centered on the statistical techniques used within that design. An important aspect of the design that must be remembered throughout the process is the verification and validation of the model itself. Verification can be defined as "the comparison of the conceptual model to the computer code that implements that conception" [Banks and Carson, 1984:376)]. In simpler words: does the model perform as intended? This question is answered on a continual basis within both the model construction and experimentation phases. A common sense list of suggestions for verification is provided in chapter ten of Banks and Carson's book. These suggestions are "basically the same ones any programmer would follow when debugging a computer program" [Banks and Carson,1984:379].

Validation is defined as follows:

> the act of determining that a model is an accurate representation of the real system. Validation is usually achieved through the calibration of the model, an iterative process of comparing the model to actual system behavior and using the discrepancies between the two, and the insights gained, to improve the model. This process is repeated until model accuracy is judged to be acceptable. [Banks and Carson, 1984:377]

One simple method of validating a model without using statistics is the Turing test. The test involves asking an "expert" in the field of interest to distinguish between real world data and the output of the model. If the expert is able to identify the model output as different, the model builder needs to work on improving the model. The Turing test can be used in an iterative fashion to fine-tune the model [Law and Kelton, 1982:338; Banks and Carson, 1984:401].

The method of comparing output to real world data can be extended into the realm of statistics through the validating of input-output transformations. This is accomplished by running the model with certain input parameters and then observing the "transformation" of these inputs into output measures of performance. These model output measures are then compared to data from the actual system using hypothesis testing on the difference in the means, generally using the 't' statistic [Banks and Carson, 1984:392-393]. "A necessary condition for the validation of input-output transformations is that some version of the system under study exists, so that system data under at least one set of input conditions can be collected to compare to model predictions [Banks and Carson, 1984:387]." The TACS is an existing system and the data obtained from several operational exercises is used for comparative purposes.

R.G. Sargent in his article, " Verification and Validation of Simulation Models," provides a list of twelve validation techniques that can be used to develope confidence in the model. Table V below is a recreation of that list.

Table V.

Validation Techniques [Sargent, 1982:162-163].

1. Face validity
2. Traces
3. Historical methods
4. Multistage validation
5. Internal validity
6. Parametric variability/ Sensitivity analysis
7. Comparison to other models
8. Historical data validation
9. Predictive validation
10. Event validity
11. Graphic displays
12. Turing tests

A few of the techniques listed have already been discussed and a thorough description of each would not be appropriate here. Refer to Sargent's article for more information. Sargent also includes an excellent bibliography covering the areas of verification and validation. Unfortunately, there are no current "cook book" approaches for the validation of simulation models. A combination of the above listed techniques is generally used, but the question of which ones to use is left to the analyst. Sargent does suggest the use of factor-screening experiments in the case of large-scale models to reduce the number of variable combinations to be tested [Sargent, 1982:167]. Pritsker states in his text that "in making validation studies, the comparison yardstick should be both past system outputs and experiential knowledge of system performance behavior [Pritsker, 1984:13]." Stated in laymen's terms, validation tests are simply checks on the "reasonableness" of the model.

As stated previously, a combination of techniques is generally used to verify and validate the model. Many times the cost and the time available to accomplish these steps play a major role in determining which technique(s) to use [Banks and Carson, 1984:402; Sargent, 1982:160]. Again, the analyst or model builder makes the final decision. Both of the following texts provide complete treatments of the verification and validations of simulation models: Law and Kelton, and Banks and Carson (both chapter 10).

Keeping on Track

With so many different techniques and suggestions to keep in mind while building the model and performing the analysis, it would be very easy to stray "off course" during a simulation effort. Having a good research design and a strict time schedule to follow helped avoid significant problems in this area. In Annino and Russell's Interfaces article, these seven reasons are offered as the most frequent causes of simulation analysis failure [Annino and Russell, 1981:63]:

1. Failure to define an achievable goal
2. Incomplete mix of essential skills
3. Inadequate level of user participation
4. Inappropriate levels of detail
5. Inappropriate language
6. Using an unverified or invalid model
7. Failure to use modern tools and techniques

Paying heed to this list helped keep the research effort on a course toward successful completion.

Summary

This chapter has described the research design for this thesis effort. A brief review of the background study of the problem preceded a description of the data collection and model construction processes. The experimental design was presented along with a discussion of the verification and validation phases. Finally, a short section covering the potential pitfalls in a simulation study was included with the aim of keeping the research "on track."

In the following chapter, the CRC model will be described as it was implemented in the TADZ model. In addition, the assumptions and changes required to use the TADZ model are highlighted.

IV. MODEL DESCRIPTION

Introduction

As stated in Chapter III, this effort began with a simple simulation model of the CRC in the SLAM language. The time spent in building this model was valuable in that much insight and understanding of the TACS operation was gained through the process. Upon finding the Transient Air Defense Zone (TADZ) model by ALPHATECH, Inc. available at FTD, it was decided to use the TADZ model to simulate the operation of the CRC within the TACS. The overall objective of the thesis did not change: to evaluate the utility of the CIS-ISS within the tactical air control system.

This chapter of the thesis will describe the simulation model at several levels. First, TADZ will be explained as used by ALPHATECH and FTD. Secondly, the TADZ model will be described as it has been modified to represent a U.S. tactical scenario at a basic level. Thirdly, a central European type scenario will be explained as it has been implemented in TADZ. The final portion of the chapter will list the changes and adaptations that had to be made in TADZ in order to model a tactical scenario and implement the identification feaure required to analyze the CIS-ISS.

Description of TADZ.

The TADZ model was developed by ALPHATECH to model the command, control, and communication process (C^3) within the Soviet air defense system. Essentially, the purpose of the Soviet air defense system is the same as the TACS, that being to provide a means of detecting, reporting, and prosecuting hostile air threats. In more analytical terms, "the goal of the air defense C^3 system is to maximize the use of scarce resources in bringing them to bear against penetrating threats [Larson and Vane, 1985:24]."

In TADZ, the enemy penetrators are the customers or entities flowing through the air defense zone and the defense assets (radars, command centers, SAM's, and fighters) are the servers. Since TADZ is essentially based on queuing theory, there are many factors within the C^3 system that can be anlayzed. Response (service) times, arrival rate of the threats, and the available resources to prosecute the penetrators are just a few of the factors that can be varied and analyzed using the TADZ model. TADZ not only models the "external" process of detecting the penetrators and prosecuting them, but it also models the "internal" decision making or command and control (C^2) process within the air defense system. Some examples of the servers in the internal system are the communications links and the track report .pa selection which operate during the transfer of information or messages between the elements of the C^2 system [Larson and Vane, 1985:26-27].

The overall emphasis in designing TADZ was "on message routing doctrine and C^3 decision making logic, while providing enough detail in the surveillance and prosecution modeling to capture the essential aspects of these phases of air defense [Merriman et al, 1984:14]." Because of this emphasis in the building process, the TADZ model was well suited for investigating the CIS-ISS question in a tactical environment. Despite the differences between the Soviet and U.S. command and control doctrine, this model can represent the "network" connectivity in a myriad of ways. In short, the TADZ model is very flexible.

Resource Requirements. "TADZ is implemented using FORTRAN-77 and the simulation language SLAM (simulation language for alternative modeling) [Merriman, 1985:52]" The actual FORTRAN coding was developed on a VAX 11-780 using the VMS operating system. The files necessary to run TADZ require approximately 4.5 megabytes of disk space just for storage of the working files. Executing the model requires approimately 10 megabytes of virtual address space.

The FORTRAN coding, which is the bulk of the TADZ model, consists of over 280 FORTRAN files. Each of these files is a subrout.ne that is compiled into a larger object file which makes up the actual executable code. Changes in the decision logic required modifying a few specific subroutines and the interfacing subroutines (those files that call or are called by the changed subroutine). [Merriman, 1985:2-3]

TADZ Modeling Approach

The TADZ modeling process can be broken down into the following catagories of "sub-models": situational models, organizational models, equipment models, process models and doctrinal models [Larson and Vane, 1985:39].

The situational models specify the geography of the scenario and the threat. The geography is detailed through such things as control boundries and the terrain. The threat is modeled through the description of the individual threat types, including speed, radar cross-section, offensive and defensive capabilities and the flight path used in the scenario.

The organizational models describe the organization of the C^3 structure and the associated connectivity between the various nodes of the network. The equipment sub-models include the modeling of the radars through the use of the radar range equation and the description of the threat prosecution assets (fighters and SAM's).

At each control or surveillance node, the processing time for information flow is modeled. Communication between the nodes of the system are assumed to be perfect in most implementations of TADZ thus far, but the model does have the capability to describe less than perfect communication [Larson and Vane, 1985:40].

The process models include the elements of detection, weapons control, and weapon allocation procedures. Also considered are such factors as equipment servicing to include fighter refueling and re-arming, as well as the reloading of the surface to air missiles [Larson and Vane, 1985:40].

As stated previously, TADZ is a model of a Soviet air defense system and, as such, models the network with Soviet doctrine throughout. Fortunately, TADZ is flexible enough through its input files that any specific differences in areas such as command policies can be explicitly treated. The latter portion of this chapter will explain the modifications that had to be made to model the identification or CIS-ISS function for this thesis. The reader is referred to the TADZ user's guide for a more detailed description of the above elements [Merriman et al, 1984]. Appendix C contains a user's guide for the TADZ implementation as constructed for this thesis effort.

Model Implementation

Thus far in this description of the model, the general elements of TADZ have been discussed. An overview of the implementation or execution phases might be helpful at this point. There are three logical phases to discuss in the implementation of the model: preprocessing, simulation, and postprocessing. The TADZ User's Guide gives the following description of these phases:

> In TADZ, preprocessing and simulation are performed in the main execution module, while postprocessing capability is provided in an independent execution file. The main execution module contains a large number of FORTRAN modules,linked to the SLAM simulation language. A substantial amount of preprocessing is done before the simulation of scenario events begins. When simulation starts, control passes to SLAM, which selects the next action to be performed....by maintaining a calendar of simulation events that have previously been scheduled. Postprocessing is done in a module written in FORTRAN and PASCAL, which contains menu structures like those used in QUEB. The postprocessor reads raw data files produced by TADZ , and provides menus that allow the user to derive stastistics from this raw data and selectively display these statistics. [Merriman et al, 1984:18]

Figure 4 displays the major elements of TADZ.

The initial execution of the model begins by reading the user input data files describing the scenario to be specifically modeled during that run. These files include order of battle information for both "sides", specifications on the radars, aircraft, and missiles, and the description of the C^3 connectivity. Again, Appendix C contains more detailed implementation information on the construction of these data files. Chapter II of the TADZ User's Guide goes into much more detail on the model structure and implementation [Merriman et al, 1984:18-43].

TADZ Network Connectivity

The network configuration modeled in TADZ is quite

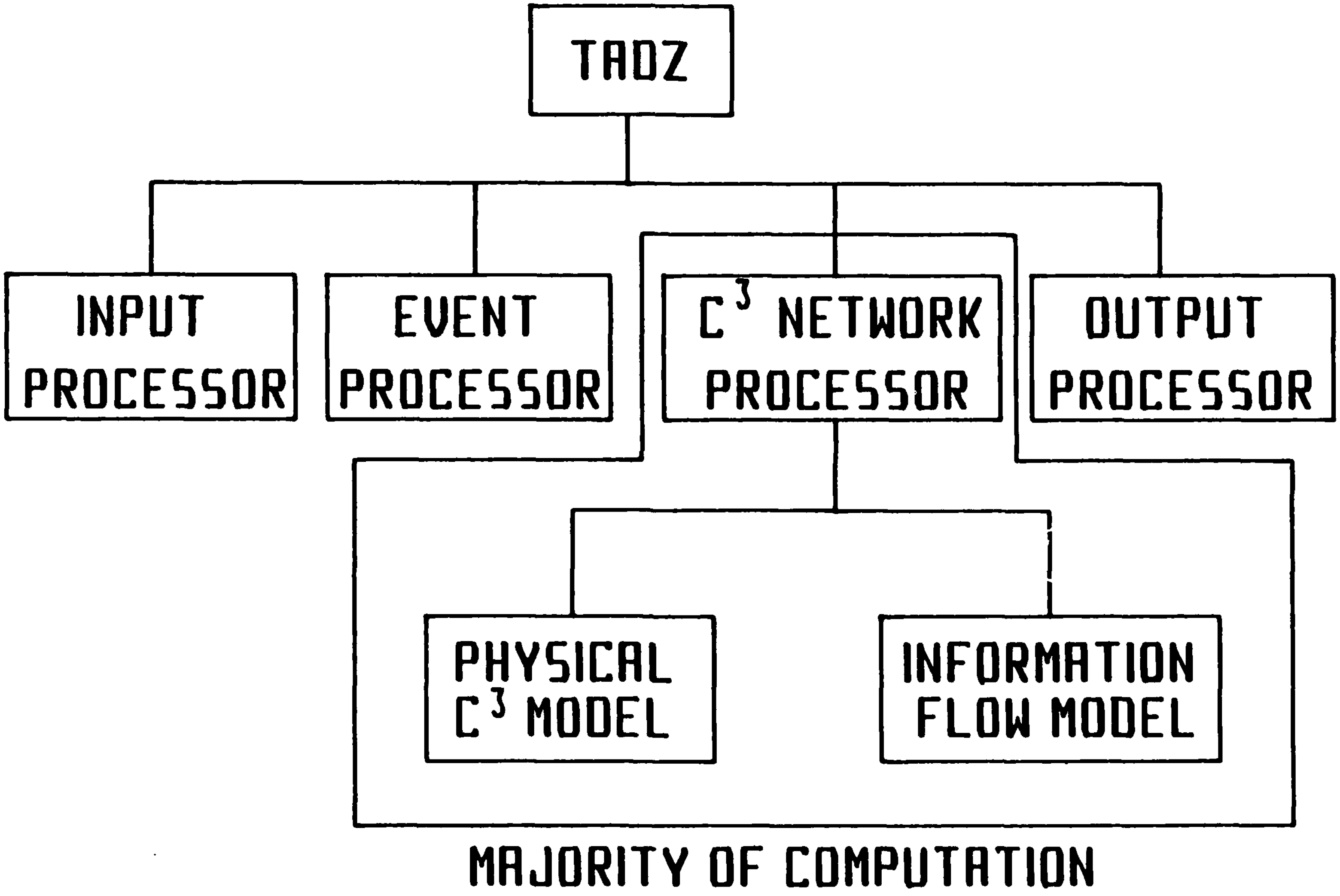


Figure 4. Major TADZ Components.

adaptive to different scenarios. Although originally set up to model the Soviet command network, it was easily reconfigured to represent the C^3 configuration of a CRC within the TACS. The TADZ documentation uses some nomenclature that may be confusing, so the following discussion will clarify and explain the labeling conventions used in TADZ along with a simple description of the CRC network and connectivity as it is represented by TADZ.

Figure 5 illustrates the TADZ representation of the Soviet style C^3 network. The labels on the various nodes in this network correspond to specific functions and locations within the surveillance and control structure. As depicted in the diagram, the flow of information follows a "heirarchical" format. The R^0 represents either an acquisition or surveillance/early warning radar. An S^0 is defined as a filtering and reporting center. The S^1 represents a sector filter center one level above the S^0 in the hierarchy. The top level of the surveillance portion of the network is modeled by the S^2 node and is known in Soviet terminology as a zone filter center. The next node in the network is the C^2 node which is the top level of the command structure or the air defense weapons operations center (ADWOC). The next level down in the command structure is the regimental/brigade command post or C^1 node. The bottom level in the command hierarchy is represented by the C^0 node or GCI/Fire Control Post. The C^0 node "controls" the fighter interceptor and SAM

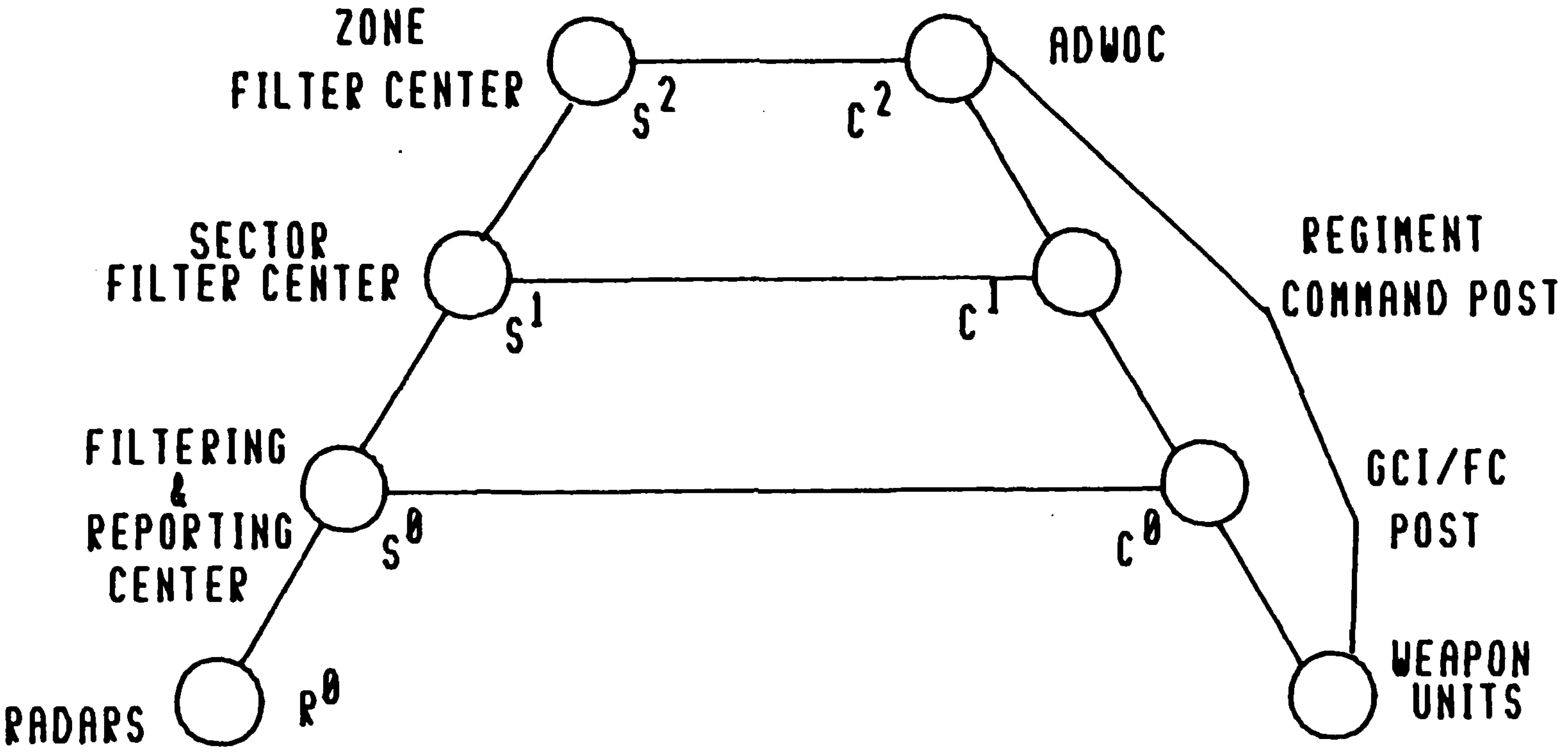


Figure 5. TADZ Network Configuration.

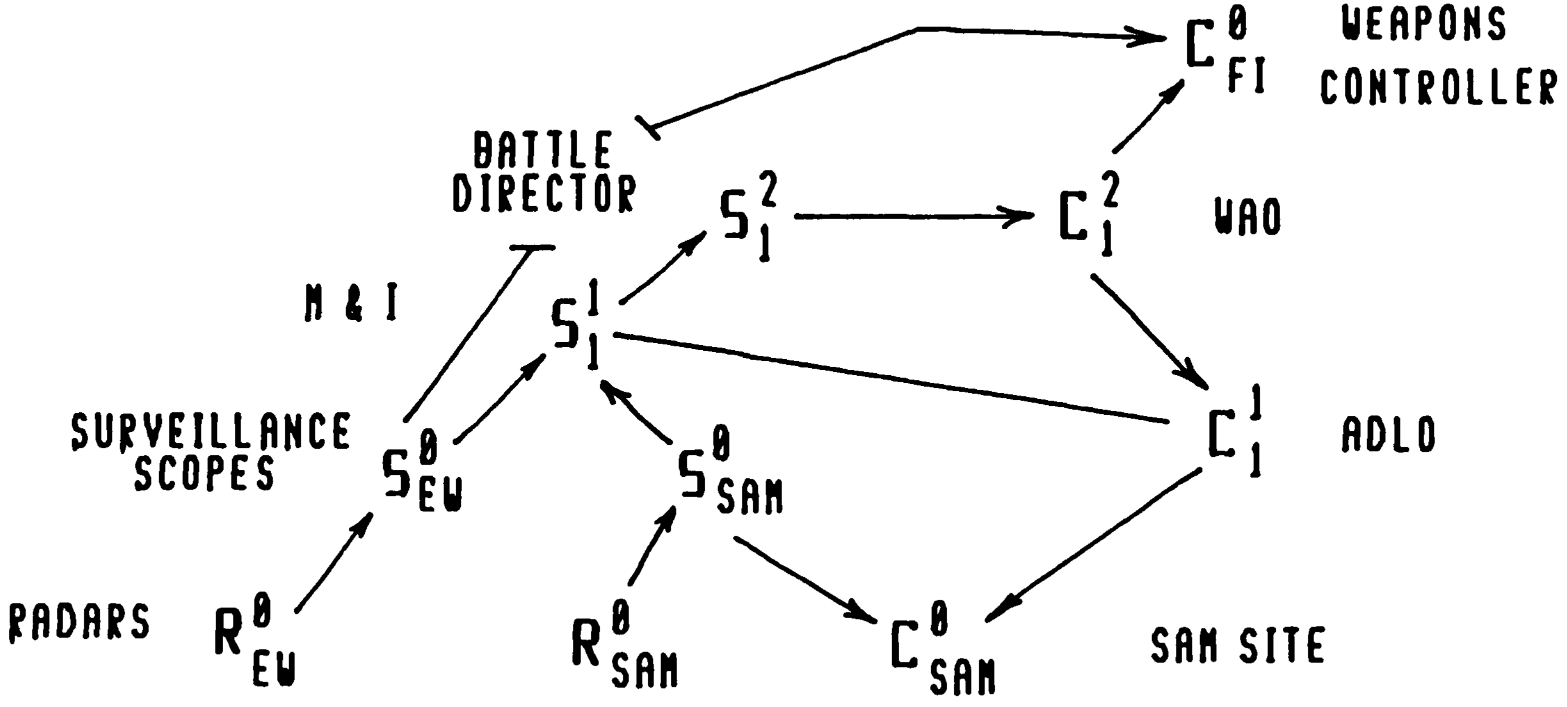


Figure 6. CRC as Represented by TADZ.

into a detailed description of the reponsibilities of each of these nodes as they exist in the Soviet air defense system, a simple network model of the CRC will be described as it is represented by the TADZ model. Figure 6 is a network diagram of the CRC within the U.S. tactical air control system.

Since this thesis is modeling a tactical air defense system, the more detailed description is included here as it relates to the functions and responsibilities within the TACS. The U.S and Soviet systems are similar in that the same type of air defense decisions and actions must be made, but the location of the nodes and the levels at which the decisions are made are generally quite different. Table VI explains the functions and locations of the C^3 nodes for the CRC/TACS as represented with the TADZ nomenclature.

For a review of the various functions within the CRC, refer to TACR 55-44 or the background section of Chapter 1 [TACR 55-44, 1985]. The detailed specifications for all these nodes are listed in the input files for TADZ. Appendix B gives instructions and examples on how to build the required files for implementing TADZ with a particular scenario.

Table VI.

TADZ Description of the TACS.

TADZ Node	TACS Function / Location
R_1^0	TPS-43 (EW radar) / CRC position
S_1^0	SSO (Scope Surveillance Operators) / within CRC
S_1^1	M&I (Movements and Identification Operators) / within CRC
S_1^2	Battle Director / within CRC
C_1^2	WAO (Weapons Assignment Officer) / within CRC
C_1^1	ADLO (Air Defense Liason Officer) / within CRC
C_1^0	Fighter Weapons Controller / within CRC
R_2^0	SAM Acquisition Radar / location of SAM battery
S_2^0	SAM Fire Control Battery / " "
C_2^0	Fire Controllers / " "

Enhanced Scenario

The CRC and TACS network just described was of a very simple air defense organization. A more detailed or expanded scenario was required to accurately model the TACS in a valid setting. This study as described in Chapter 1, is based on a Central European scenario with a massive conventional attack from the WARSAW Pact forces. The air defense in this region is essentially divided into two zones (4 ATAF, 2 ATAF). The scenario chosen for this simulation models a "generic" U.S. air defense zone [Spaeth, 1985]. Figure 7 is a depiction of this scenario. Again, this is a generic scenario and future users could easily build a simpler or more advanced simulation by simply rebuilding the TADZ input files. This scenario includes essentially one early warning/ground controlled intercept (EW/GCI) radar, three combat air patrols (CAP) points (COFI1 - COFI3), four SAM batteries (with associated equipment, MI1 - MI4), and one CRC. The lethal engagement zones for the SAM batteries are indicated by the hashed circles. The TADZ network representation is much more complicated than the simple CRC model described earlier. Figure 8 is the network depiction of the final CRC scenario.

As stated earlier, this scenario could be enhanced fairly easily. AFHRL's envisioned use of the model should not require any drastic changes in the scenario or the other parameters within the FORTRAN coding of TADZ. The information in Appendix B will enable the user to make the required changes to the scenario. Any deeper excursions into coding changes would require a more in-depth study of TADZ

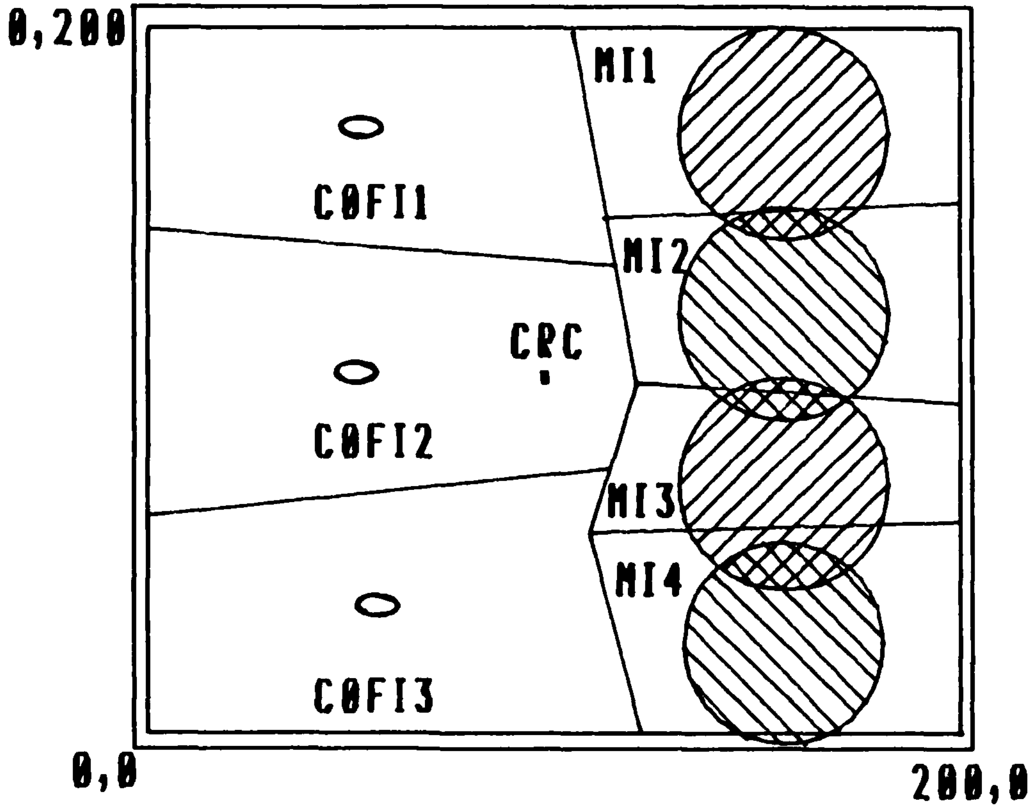


Figure 7. Thesis Air Defense Scenario.

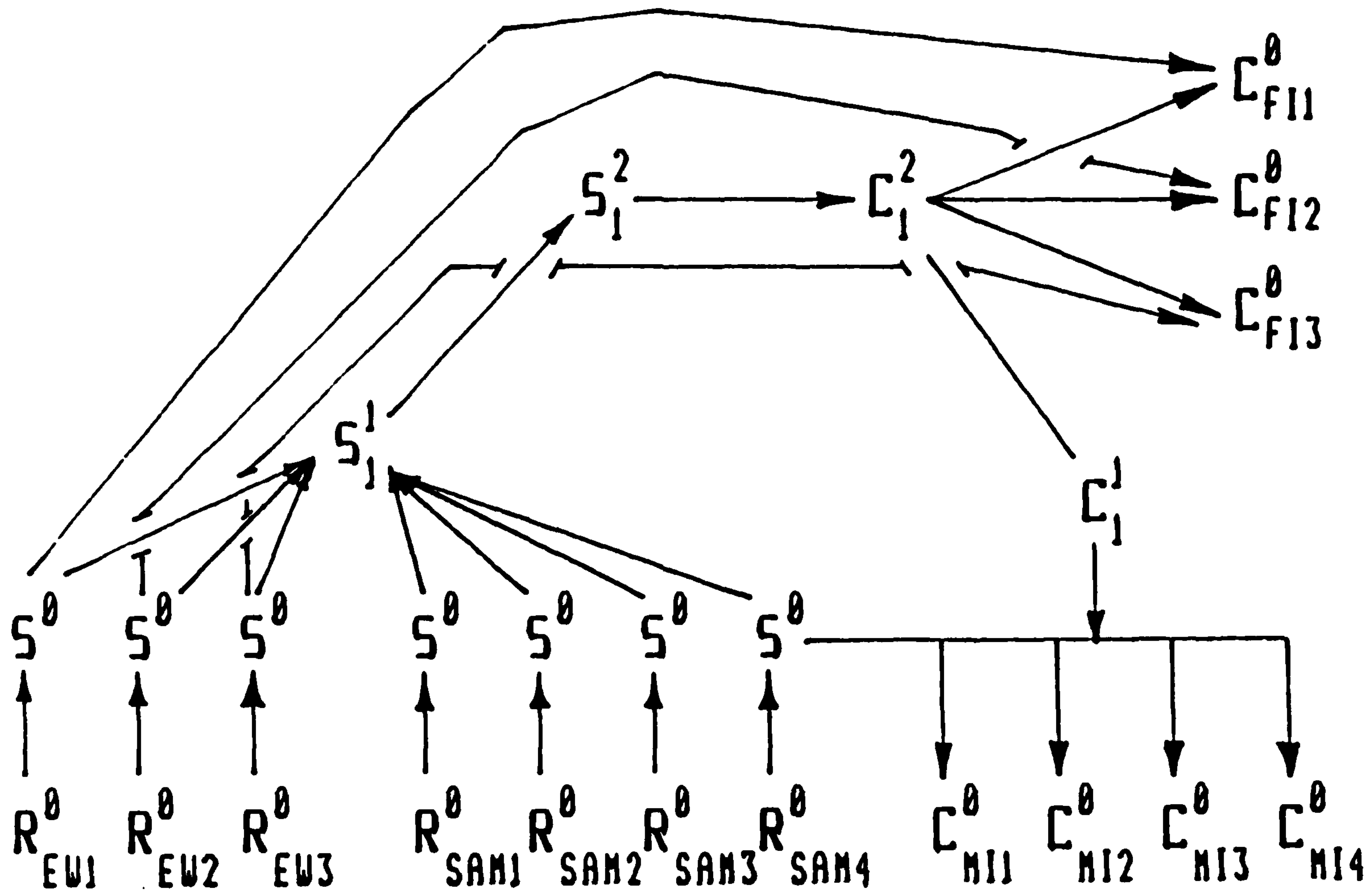


Figure 8. TADZ Depiction of Final Scenario.

and its implementation. The TADZ instruction manuals (Vol I, II, III) available through FTD/TQC or ALPHATECH, Inc. are recommended for this purpose. The next section of this chapter will deal with the FORTRAN changes required in TADZ to model the identification function in the tactical scenario.

TADZ Modifications

In order to implement the functioning of the CIS-ISS in the TADZ model, three major modifications were required. The three modifications are required to emulate the realistic flexibility of operations available in the TACS and not available in the TADZ model. The three operational functions not in the TADZ model are: (1) the ability to decide not to prosecute a penetrator, (2) the ability to distinguish among penetrators (friend from foe), and (3) the ability to perform an identification intercept.

First, the model needed the capability to not prosecute a penetrator. The existing TADZ logic assumed that all radar detections were hostile aircraft attempting to penetrate the air defense zone. Since all penetrators were assumed to be hostile, the air defense system made an attempt to destroy each penetrator with the assets available. This assumption is not valid for the scenario employed to test the CIS-ISS. The CIS-ISS was tested for its ability to assist in the identification process. This identification process will not only distinguish friend from foe but will also provide the probable type. The system needs the ability to not prosecute a detected aircraft when the aircraft is identified as friendly or for some other command decision. This selective prosecution capability was not provided in the original TADZ coding.

The second major modification of TADZ required a change to the model so that an identification code could be attached to each penetrator. This modification reflects the identification friend or foe, selective identification feature (IFF/SIF) present in the existing TACS. The previously modified selective prosecution feature could then be selected based on the information contained in the identification code. The actual identification codes used in the model are detailed below.

The third feature added to the TADZ model was the identification intercept. Based on the existing operating orders or the rules of engagement, a penetrator may require a visual identification prior to execution authorization. This type of execution requires the deliberate withholding of the authorization for missile engagement and the explicit use of fighters. In addition, the fighters are restricted from using long range weapons against the penetrator prior to the pilot making a visual identification of the penetrator. Penetrators which cannot be specifically identified as friend, foe or neutral require this type of identification intercept.

Additionally, the TADZ model used the uniform distribution to model the average service times for message processing at the TADZ nodes. Since the analysis of the Eglin test data showed that these processing times actually fit the exponential distribution, the message processing was changed from uniform to exponential in the subroutine SERVTIME.FOR.

TADZ Entities. Externally, the model treats the penetrators entering the air defense zone as the customers to be served. Internally, the entities that flow through the TADZ model are the command and control messages which would normally be distributed through an air defense system. The processing of the messages within the SLAM network represents the servicing of the penetrator customers. In order to change the TADZ network to implement the modifications required for the CIS-ISS function, two methods could have been used. First, the messages entities could have been modified to include the identification of the penetrator. Secondly, the processing of the messages could have been changed so an identification code would indicate whether the penetrator was friendly and whether or not the air defense system should prosecute the detected aircraft.

Because the timing in the TADZ model is based on time elapsed while processing the messages generated by the various nodes in the air defense system, it was important to ensure that the selective prosecution feature did not affect the normal message flow. For instance, just because a penetrator was identified as friendly does not mean that the messages concerning the aircraft could be eliminated from the system. In fact it is conceivable that the processing load for messages on friendly aircraft could interfere with the message processing for the hostile aircraft. Therefore, in order to preserve the current flow of messages, the selective prosecution feature was implemented at the last processing node possible.

Selective Prosecution

The selective prosecution feature was implemented in the following manner. The TADZ logic requires two message events to be true before an execution node (fighter or missile C^0) will prosecute a penetrator. The first message the C^0 prosecution node checks for is the radar detection report. TADZ logic requires that each individual C^0 execution node have a detection report from an S^0 node that is directly connected to the C^0 node. The detection reports routed via the surveillance side of the air defense system are not sufficient for prosecution, the report must be direct from the S^0 to the C^0. The second message event required prior to initiating a prosecution of a penetrator is an assignment (or alert) message. In other words, the surveillance section needs to detect the hostile penetrators and identify them as hostile. Then, the weapons control side of the air defense system will make the decision as to which weapons controller (working a specific CAP point) or Hawk missile battery will engage the penetrator. When both (detection and assignment) messages are present at a C^0 node, TADZ then 'matches' the detection and assignment messages and attempts to prosecute the penetrator if the assets are available. For this reason, the C0MATCH subroutine was modified to not file detection reports at C^0 nodes for selected penetrators based on their identification code.

C0MATCH. The actual FORTRAN selective prosecution code modifications made to TADZ were done in the FORTRAN subroutine called C0MATCH.FOR. It is in this subroutine that the system checks to see if both types of the required messages for prosecution exist at the specified C^0 node. The subroutine returns a logical value of true or false for the detection and assignment messages based on the status of the messages waiting for processing at the node. To avoid affecting normal message processing of TADZ, the messages themselves were not changed. Instead, the identification of the penetrator is checked to see if the identification does not require prosecution (ID code 5 or higher). If the penetrator does not require prosecution the single consolidated assignment message produced by C0MATCH is not placed in an assignment queue. However, the penetrator is still reported through the surveillance side of the CRC. This surveillance report will ensure that event though the system cannot prosecute the penetratore, the model still maintains the penetrator in the target data base.

An IF THEN statement checks the identification code attached to the penetrator and fails to determine a queue for the assignment message at the C^0 for selected values of the identification code. The following identification codes were used in the COMATCH subroutine:

1 = Hostile identified as Unknown

2 = Friendly identified as Unknown

3 = Freindly identified as Hostile

4 = Hostile identified as Hostile

5 = Friendly identified as Friendly

6 = Hostile identified as Friendly

7 = Deliberate non prosecution

8 = New code for unknown friendly

If the identification code is 5, 6, 7, or 8, the COMATCH subroutine does not place the assignment message in a queue which would enable execution. Since the added code is at the end of the subroutine, normal message processing has already been accomplished. The failure to place the assignment message in a queue only prevents the C^0 node from actually beginning prosecution on selectively identified aircraft.

Identification Feature

In order for the selective prosecution feature to work, the individual penetrators should have the identification code embedded in the message entities that flow through the TADZ network. However, the queues that represent the air defense system nodes are based on the different penetrator types. The messages which are to be processed are placed in the queues based on whether the messages are first detection reports or continued reports. Thus, the number of queues attached to each node which represents a portion of the air defense system is eight. There are two queues for each penetrator type (maximum of 4), one for first reports and one for continued reports. The creators of TADZ also allowed a ninth node in the system for expansion. Since the identification codes are not limited or tied to the penetrator type, the queues were not used to manipulate or use the identification code. In addition, the computing overhead that would be required to pass the attribute array each time a C^0 node evaluates a penetrator would make such a method of implementing the identification feature cost prohibitive in execution time.

The TADZ FORTRAN code provides a means to collect the data on the penetrators as they progress through the system. The penetrators and their identification codes are initialized in the PENINT.FOR subroutine and then passed to each routine that needs the data in the PENIDENT.INC common block definition. The subroutine that prints the final output file is PRINTPEN.FOR. This subroutine prints output which contains the penetrator number, the penetrator path, and the times of first detection, first assignment, first engagement, destruction, and identification code. The following paragraphs describe the changes made in each of these subroutines.

PENIDENT.INC. The labeled common block contains the identification information on the penetrators. The specific identification code can be referenced using the array PEN_IDENT (I). The index of the array PEN_IDENT is the penetrator number so that each value of the array is tied to a specific penetrator.

PENINT.FOR. The FORTRAN subroutine PENINT.FOR actually sets the identification codes used in the model. The identification code is determined using a random draw from a uniform distribution. The identification code is then set using IF THEN ELSE statements and predetermined probability values. The different levels or capabilities of the CIS-ISS to perform the identification feature can then be varied for analysis by changing the probabilities in this subroutine. Figure 9 shows a decision tree for one set of factor levels for accuracy (.99) and capability (.06). Recall that 10 percent of the penetrators were assumed to be friendly, and the remaining 90 percent were hostile.

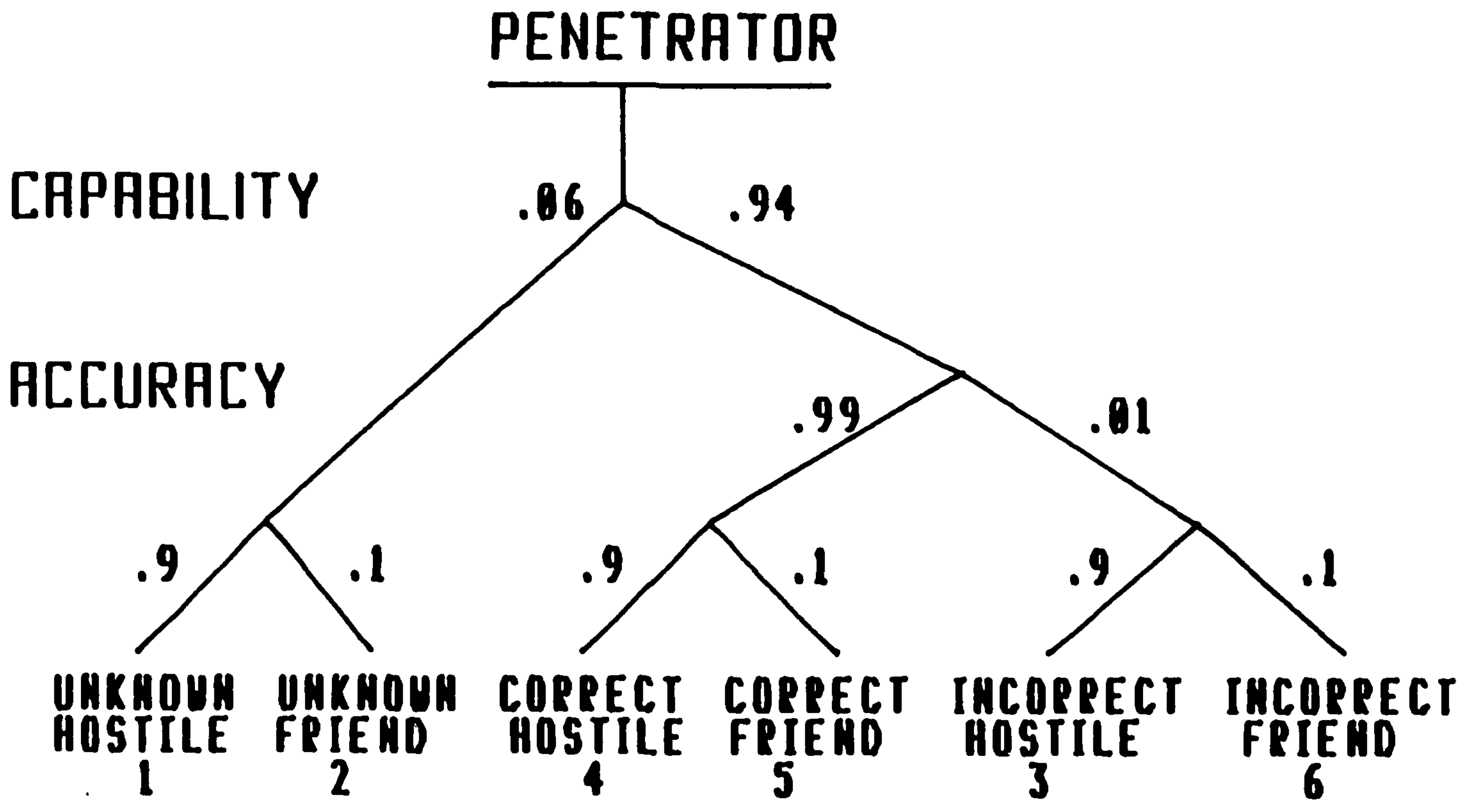


Figure 9. Identification Probabilities.

PRINTPEN.FOR. This subroutine provides the output that is used by TADZ instead of the SLAM statistical collections. This output file was changed to provide a means of tracking the identification code that was generated in PENINT.FOR. This also allows the analyst to verify that the appropriately coded penetrators are not executed. The code changes added the penetrator identification code to the existing output file for verification by the analyst.

Identification Intercept

Emulating an identification intercept within the TADZ model required the ability to designate the penetrator for execution by fighters only. In addition, the fighters must not use long range weapons, rather they must conduct a visual intercept. Once identified, the model needed the ability to either abort the intercept or destroy the penetrator when the visual identification was completed.

The actual code changes required to implement an identification intercept was made in three files. The SODISPTCH.FOR was further modified, along with the subroutines which control the release of long range and short range weapons (LONGFIRE.FOR and SHORTFIRE.FOR).

COMATCH.FOR. The coding in this subroutine was given an expanded IF THEN structure to sequence through both the identification code and the node labeling structure to ensure that a valid assignment message is placed only in the message processing queue for C^0's identified as a fighter C^0. The actual coding assumes foreknowledge of the number of fighter C^0s in the scenario. Changes in the number of fighter C^0s would require changes to the code and recompiling of the TADZ model. See appendix B for the detailed procedure to accomplish this straightforward modification and a listing of the actual code changes implemented.

LONGFIRE.FOR. The decision logic required to fire a long range missiles from a fighter interceptor is contained in this subroutine. The FORTRAN coding was modified to test the identification code prior to expending any missiles. If the identification code was UNKNOWN (1 or 2), the decision logic was automatically switched to SHORTFIRE.FOR to emulate the requirements for a visual identification.

SHORTFIRE.FOR. This subroutine contains the decision logic for fighter prosecution against penetrators at short range. An IF THEN statement was used to test the identification code. If the penetrator was identified as friendly (2), the intercept is coded as a missed intercept, the fighter prosecution is canceled and the penetrator is given a new identification code (8). If the penetrator is identified as hostile, the penetrator is destroyed.

Validation and Verification

Validation. The TADZ model provided by the Foreign Technology Division (FTD) has already been verified and validated for use at FTD. It is currently being used in studies conducted at FTD. Therefore, if the code changes can be shown to have no effect on the original processing of the model, the model is still a valid model. In addition, the output results and penetrator prosecution were reviewed by air defense experts at the Human Resources Labratory for face validity. No obvious discrepancies were detected.

Verification. Each change of the TADZ FORTRAN coding needed to be checked to ensure the original message processing was not affected by the coding changes added for the identification feature. The code was verified by comparing the following test cases against a base case (before any code changes) running identical scenarios with changes only in the identification codes used. First, all penetrators were initialized with a friendly identification code to ensure that none of the penetrators were attacked. All the penetrators passed through the system without being assigned to a C^0 for prosecution while they were still detected at the original times reported in the base case. Second, all the penetrators were initialized with an unknown/hostile identification code to ensure that only FI C^0s attack the penetrators. While the assignment and destruction times were different, this was to be expected as the surface to air missile systems were not used. Next, the identification code was changed to unknown friendly. As expected, the detection and assignment times reported by TADZ were the same as in the second case. The fourth case consisted of assigning all hostile identification codes to the penetrators. Since all of the penetrators were identified as hostile, the detection, assignment, and destruction time should be identical to the base case -- which they were. Since there was no reported differences in the TADZ output between the original coding and the modified coding, the code changes were verified as having not adversly modified the original TADZ message processing. Thus the model changes have not affected the validity of the original model.

Summary

Chapter IV has provided a brief overview of the TADZ model as provided by the Foreign Technology Division. A description of the available system nodes and their use in representing the tactical control system processes has also been explained. In addition, the required modifications to implement a selective prosecution function, an identification function, and an identification intercept feature in the TADZ model were described. The detailed files, code changes, and user manuals required to implement the TADZ model are contained in appendices A, B, and C. A brief description of the verification and validation procedures was also described. In the next chapter we will provide a detailed description of the experimental design and the various input levels used for data collection. A screening analysis will identify which factor levels have significant effects on the model. An analysis of the effects due to changes in the input levels will also be presented.

V. ANALYSIS OF RESULTS

Introduction

This chapter des^ ibes and explains the results of the experimentation with the modified TADZ model. The experiments were chosen to answer the research question as to whether or not the CIS-ISS would be a valuable aid in the identification process within the CRC and TACS. The experimental design will be reviewed, the measures of merit restated, and the major findings will be discussed. Some sensitivity analysis was accomplished along with some excursions from the original scenario. These will be included at the end of the chapter along with an interpretation of the experimental results relative to the research question.

Factors in the Analysis

With such a complicated simulation model there were many input variables to consider as factors for the experiment. Several of these variables such as number of fighter interceptors, number of SAM's, types of penetrating aircraft, and the routing of the penetrators were considered to be scenario dependent and, thus, eliminated from the list of factors to be varied by using a fixed scenario. Since the key issue in this effort is whether or not CIS-ISS improves the identification process, the problem is essentially one of comparison against a standard. With the objective of answering this question, three factors were chosen based upon their power to "describe" the identification function. These factors were identification capability, accuracy, and time to identify. A discussion of these factors along with their associated levels follows. Table VII lists the factors used.

Table VII. Factors and Levels.

Factor	LO	MED	HI
1. Capability (percent unknown)	10	6	2
2. Accuracy (percent)	80	90	99
3. Identification Time (seconds)	10	60	112

Capability. The capability of the identification system, both manual and automated, is defined in this thesis as the ability to sense a target and assign an identification to the penetrator. Using this definition, the capability of the system can be measured by observing the percent of "unknown" labels assigned compared to the total targets processed. The three levels selected for this factor were 10, 6 and 2 percent. Since the unknown identification imposes restrictions on the ability of the air defense system to select the appropriate asset for prosecution, an unknown level of "one out of ten" was a reasonable worst case. The level was increased by 4 percent in two increments to increase the capability until the system was five times better, or one out of fifty. The system could provide a specific identification code 90, 94, and 98 percent of the times respectively. These levels are set in the PENINT.FOR subroutine of the TADZ code. (See Appendix A for the implementation of the levels used in the experiment.)

Accuracy. The accuracy of the identification was another important factor to consider. It should be clear that the ability to identify foe from friend is a critical and highly desirable characteristic of an identification system. The initial estimates on CIS-ISS performance advertise a 99 percent accuracy [Perini, 1985:81] Given a detection, the system is reportedly capable of identifying the correct type of the aircraft 99 percent of the time. This variable is also set in the PENINT.FOR subroutine in TADZ. Using the advertised accuracy of CIS-ISS as an upper bound, the following levels were chosen to provide a wide range of operational conditions: low = 80 percent, medium = 90 percent, and high = 99 percent. These levels were considered to reasonably cover the possible accuracy range of both the CIS-ISS and the manual systems.

Identification Time. The final level chosen for the analysis was identification time. One of the perceived advantages of the automated system over the manual system is the savings in time through a faster identification. One of the difficulties in choosing levels for this factor stems from the lack of unclassified data and also the lack of operational tests conducted at a threat level high enough to simulate the maximum wartime system load. The higher (slower) limit for this factor was considered to be 112 seconds per track. This level is based on data reduced from the CIS-ISS operational test at Eglin AFB in May of 1984 [Preidis, 1985]. The medium level was selected as 60 seconds, half the maximum allowable time. A low level of 10 seconds was arbitrarily selected as a reasonable lower limit on the processing speed of the CIS-ISS. These levels were also compared against some operational test and evaluation data on the CRC performed in 1971 which showed an identification time average of 65 seconds [CRC OT&E, 1971:85]. The actual values used were input as the "first service time" within the S101.DAT input file for the modified TADZ model. The code reads these values as the means of exponential distributions. Appendix D lists the goodness of fit tests and results of the data reduction and distribution analysis for several of the model inputs including the high value of the identification time.

Measures of Merit

There were two measures of merit or performance selected to evaluate the differences between the various combinations of the input factors. The first measure of merit was the number of hostiles that get through the air defense zone without being engaged and "killed" by either the SAM's or fighters. The second measure was the number of fratricides that occurred during each run of the simulation. These two measures were deemed to capture the essence of what an identification decision aid should bring to the tactical air control system -- the ability to destroy more enemy aircraft while destroying fewer of our own.

Analysis of Variance

With three factors at three levels each, the resultant experiment was a 3^3 design requiring 27 simulation runs for each replication desired. See Figure 10 for a three dimensional representation of the 3x3 design. Since the factor levels were not randomly picked, this analysis is a fixed effects model and consequently the results cannot be extrapolated outside the factor levels of the design [Montgomery, 1984:44].

Due to the size of the experimental design, it was decided to limit this initial experiment to only one replication. This decision required an assumption that the three way interaction was negligible in its contribution to

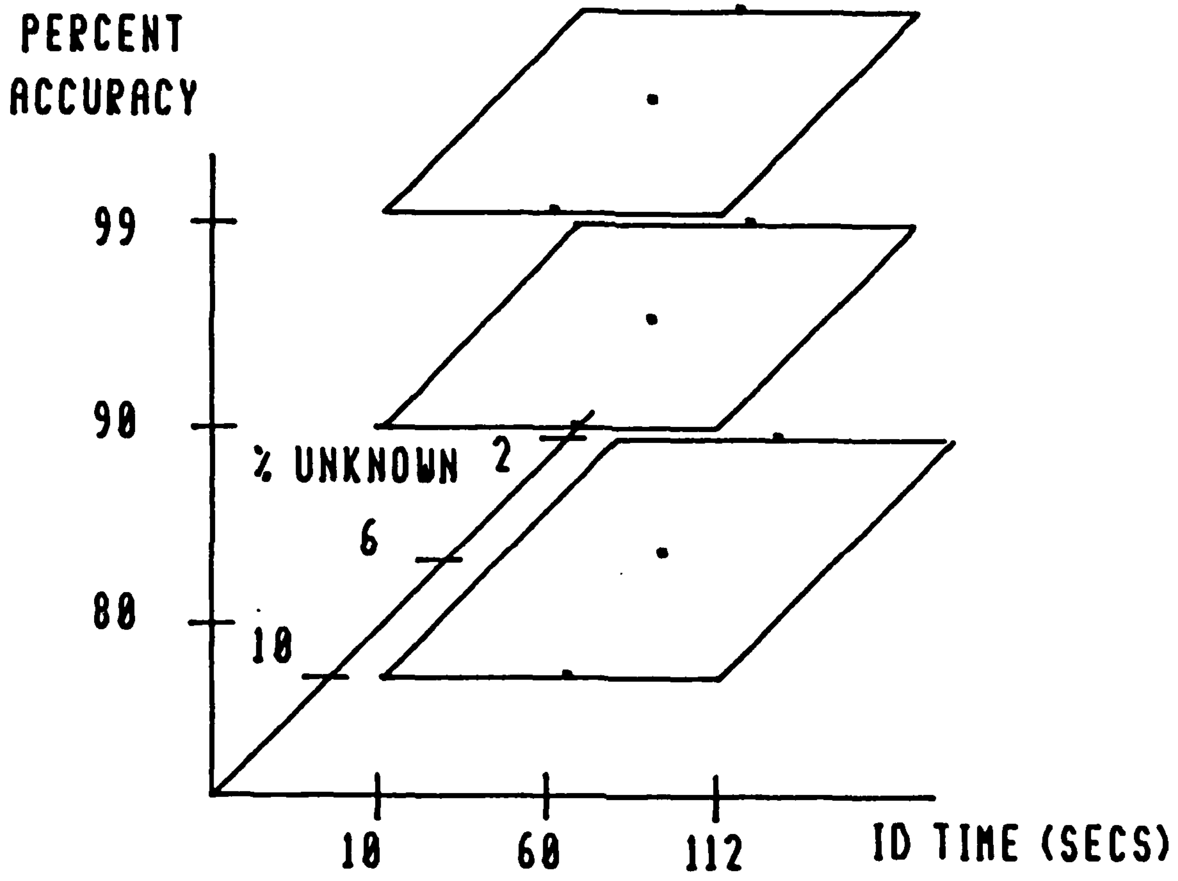


Figure 10. Graph of 3x3 Design.

explaining tne variance. An expanded "Tukey's additivity" test was performed on both of the measures of merit to verify the assumption of negligible three level interactions which is required to justify the use of one replication in the analysis of variance [Kirk, 1982:250-253]. The additive model (no three level interaction) assumption was verified at alpha levels ranging from 5 to 25 percent as suggested by Kirk.

The ANOVA for the number of successful penetrators is contained in Table VIII. Note that for the first measure (number of hostiles through), identification time and accuracy were both significant at the 5 and 10 percent levels indicating a definite contribution to the variance. The capability factor was found to be insignificant at both 10 and 5 percent levels. A brief discussion of the effects of each of the factors is presented below.

Accuracy. As the graphs in Appendix D indicate, as the accuracy of the system increases, the number of penetrators that get through the air defense zone decreases. This effect of accuracy on the number of hostiles killed is intuitively appealing. The more accurate the identification, the greater the number of hostiles that should be killed. This is because fewer hostile aircraft get through the air defense zone because they are miss-identified as friendly.

ID Time. However, the effect of identification time on the first measure of merit was different than expected. The

Table VIII. ANOVA for Penetrators Through

Variate Factor	Sum of Squares	Degrees of Freedom	Mean Square	F Statistic
i: IDtime	2968.2	2	1484.1	22.38[ab]
c: Capability	12.6	2	6.3	0.10
a: Accuracy	5936.8	2	2968.4	44.77[ab]
ic	371.1	4	92.7	1.40
ia	1362.2	4	340.2	5.14[ab]
ca	301.1	4	75.2	1.14
ica: Error	530.4	8	66.3	
Total	52801.3	1		

a = significant at 5% b = significant at 10%

number of hostiles through was less for the longer ID times! Upon examining the model outputs for network message flow, it appeared that as the system processes more and more detection messages, the M&I section begins to backlog. However, the system does not backlog forever. If the TADZ model determines that normal message processing (through the M&I section) will prevent penetrator prosecution, the model will allow the individual C^0s to begin prosecution if that is the only means to prevent the penetrator from successfully penetrating the air defense zone. As the ID time is extended, the backlog occurs earlier and the system defaults sooner. As the model defaults at an earlier time, the C^0s begin to prosecute every penetrator within its operating area, thus accounting for the fewer successful penetrators. The default TADZ model prosecution mode also does not resolve dual prosecution for those C^0s with overlapping coverage. The model reflects the actual operational processes in this regard. The weapons assignment section will not allow penetrators to exit the system without prosecution just because of a backlog in the movements and identification section.

There is also some indication that the use of speed of identification as a factor may have been a less than optimum choice as a factor variable. Using time as a criteria for ID does not preserve the independence between the ID time factor and the tactical decision time for employment of assets. Chapter VI contains a more detailed discussion of this point.

Capability. The level of the capability or percent of unknowns was found to have no significant effect on the model outputs. The increase in the number of unknowns will cause an increase in the number of engagements required for fighter interceptors and fewer engagements for the SAM sites. A closer examination of the output data revealed that the number of SAM kills remained constant over all factor levels in the model. The number of wasted assets would be the sorties that engaged unknown friendly aircraft. Since only 10 percent of the penetrators are friendly and the number of unknowns ranged from 7 (320 x .02) to 32 (320 x .10), the

Table IX. ANOVA for Fratricide

Variate Factor	Sum of Squares	Degrees of Freedom	Mean Square	F Statistic
i: IDtime	8.296	2	4.148	0.61
c: Capability	7.629	2	3.814	0.56
a: Accuracy	235.185	2	117.592	17.26[ab]
ic	22.370	4	5.592	0.82
ia	13.481	4	3.370	0.49
ca	8.148	4	2.037	0.30
ica: Error	54.518	8	6.814	
Total	370.37	1		

a = significant at 5%   b = significant at 10%

number of penetrators on which to waste assets would vary between less than one (0.7) and just over three (3.2). This range is over a small enough interval to have no significant effect on the results. This would also tend to indicate that the CIS-ISS would need to fail to identify 10% or more of the penetrators for this factor to be significant.

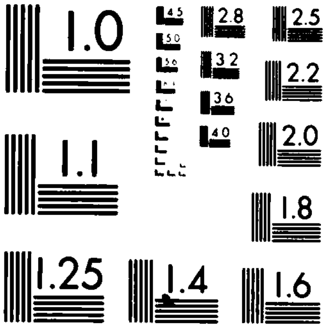
Fratricide. The ANOVA results for the second measure (fratricides) showed that the only significant factor contributing to the output measure was accuracy. Table IX shows the ANOVA for fratricide with the accuracy "significant" at both the 5 and 10 percent levels. This result is explainable as the only friendlies that would be killed are the aircraft that are identified incorrectly as hostiles. Since this result depends solely on the identification accuracy (in this model), the results are as expected.

Interaction graphs for both measures of merit were constructed for all possible combinations of the three factors at all three levels. The significant graphs are displayed in Figures 11 and 12. Figure 11 shows three graphs with capability being held constant at one level for each of the grapns. Note that as accuracy and time increase the graphs decline to the right showing the previously mentioned decline in the number of hostile penetrators making it through the air defense zone. Figure 12, which has time held constant for each graph, shows that as capability increase for each level of accuracy there is little overall effect. These graphs further describe the trends and results just discussed. The remainder of the interaction graphs are found in Appendix D.

Sensitivity Analysis on Enemy Tactics

The initial analysis just described provided a basis for choosing the combination of factors which were used for modeling the CIS-ISS in the baseline scenario. The following factor levels were chosen to model the CIS-ISS. Accuracy was set at the 99 percent level to maintain a zero fratricide level. Capability was set at 98 percent and a 10 second ID

AD-A172 447   A SIMULATION ANALYSIS OF AN AUTOMATED IDENTIFICATION   2/3
PROCESSOR FOR THE TA..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..
UNCLASSIFIED   R C MACFARLANE ET AL. JUN 86   F/G 17/7   NL


1.0
4.5
5.0
5.6
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

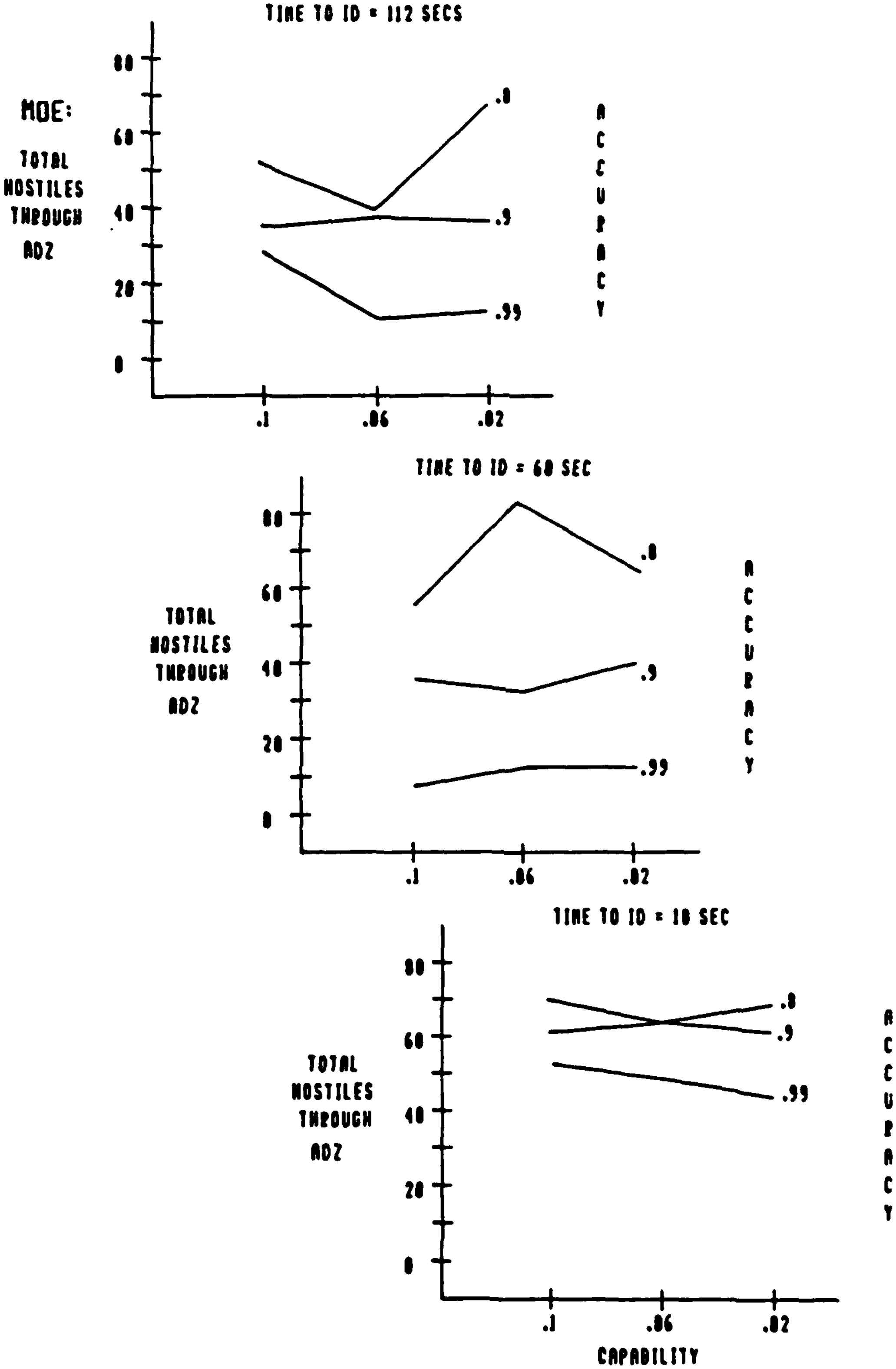


Figure 11. Interaction Graphs for Capability.

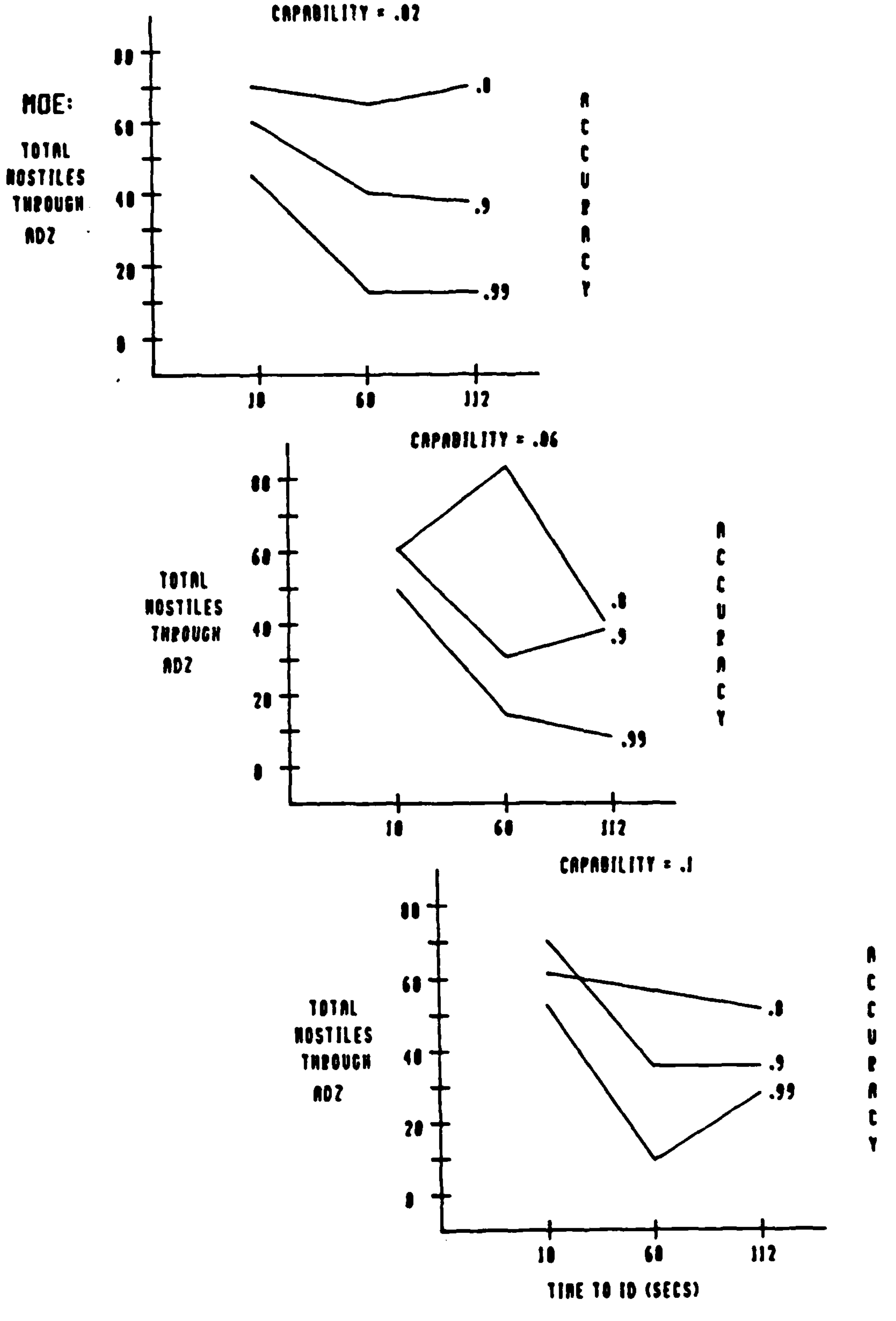


Figure 12. Interaction Graphs for ID Time.

time was chosen as the fastest processing time. With the CIS-155 factor levels at these settings, the model was tested for sensitivity to variations on enemy tactics as described in the following paragraphs.

Reduced Number of Penetrators. The original scenario used a total of 320 penetrators. This number was an upper limit due to memory limitation of the TADZ model. This limitation will be further explained in Chapter VI. A reduction of penetrators of 20 percent (256 vice 320) was input and the results were compared to the original scenario. A one way ANOVA, using a sample size of 10, indicated a significant difference in means between the two levels of penetrators using the F statistic at the 10 percent level. This difference was found to be statistically insignificant using the Duncan multiple range test at the 5 and 10 percent levels. The natural conclusion was that the number of penetrators did not change the results of the simulation. Upon examining the times of the penetrator kills, it was found that the air defense system is most effective during the start and end of the wave attack. In general, the successful penetrators are the middle of the wave while the system is occupied with the start of the wave. See Table X for the actual test data results on variations of enemy tactics.

Flight Size. The original scenario used a constant value of 4 aircraft per flight of penetrators. Spreading out the total number of hostile aircraft by reducing the flight size should make the air defense tasks more difficult. This hypothesis was tested by reducing the flight size to a constant level of 2 aircraft. With the number of replications set to 10, a one way ANOVA was performed to compare to the original scenario. The F test indicated a significant difference in cell means and the Duncan test confirmed that the smaller flight size increased the number of hostiles that "got through" the system (as hypothesized). Table X shows the pertinent statistics.

Spacing of the Penetrators. The number of seconds between the penetrator flights was also thought to effect the performance of the air defense system. The original scenario

Table X. Sensitivity on Enemy Tactics.

Scenario	No. Through (Mean)	F-stat Signif?	Duncan Test Signif?
Automated	38.0	----	----
Penetrators 256	36.9	Yes	No
Spacing 30 seconds	58.6	Yes	Yes
Flight Size 2 A/C	49.0	Yes	Yes
Path Routing	50.8	Yes	Yes

used a value of 15 second spacing between individual flights. This was considered an upper limit on compressing the arrival rate based on information from Soviet Military Power, and Coyne's article in Air Force Magazine [Soviet Military Power, 1985:103; Coyne, 1984:74]. Ten replications were run at a spacing of 30 seconds and the results were compared to the original scenario using the one way ANOVA. Both the F test and the Duncan test indicated a significant increase in the number of hostiles that penetrated the air defense zone when the spacing was increased. This result was to be expected as the fighter assets were made to work "harder" when the penetrators were more spread out. Thus the penetrators did not present such lucrative target clusters. The actual statistical results are found in Table X.

Penetrator Paths. The original scenario had 8 parallel penetrator paths running through the 100 km by 100 km air defense zone. A more realistic tactic for the enemy would be to vary the routing of the paths to increase the difficulty of the tracking and intercepts by the air defense assets. The paths were changed to reflect such a tactic and the ANOVA was again used to compare the results with the original routing. See Figures 13 and 14 for a comparison of the two scenario penetration paths. Both the F and Duncan tests indicated a significant increase in the number of penetrators not killed when the paths were made more complex. Table X

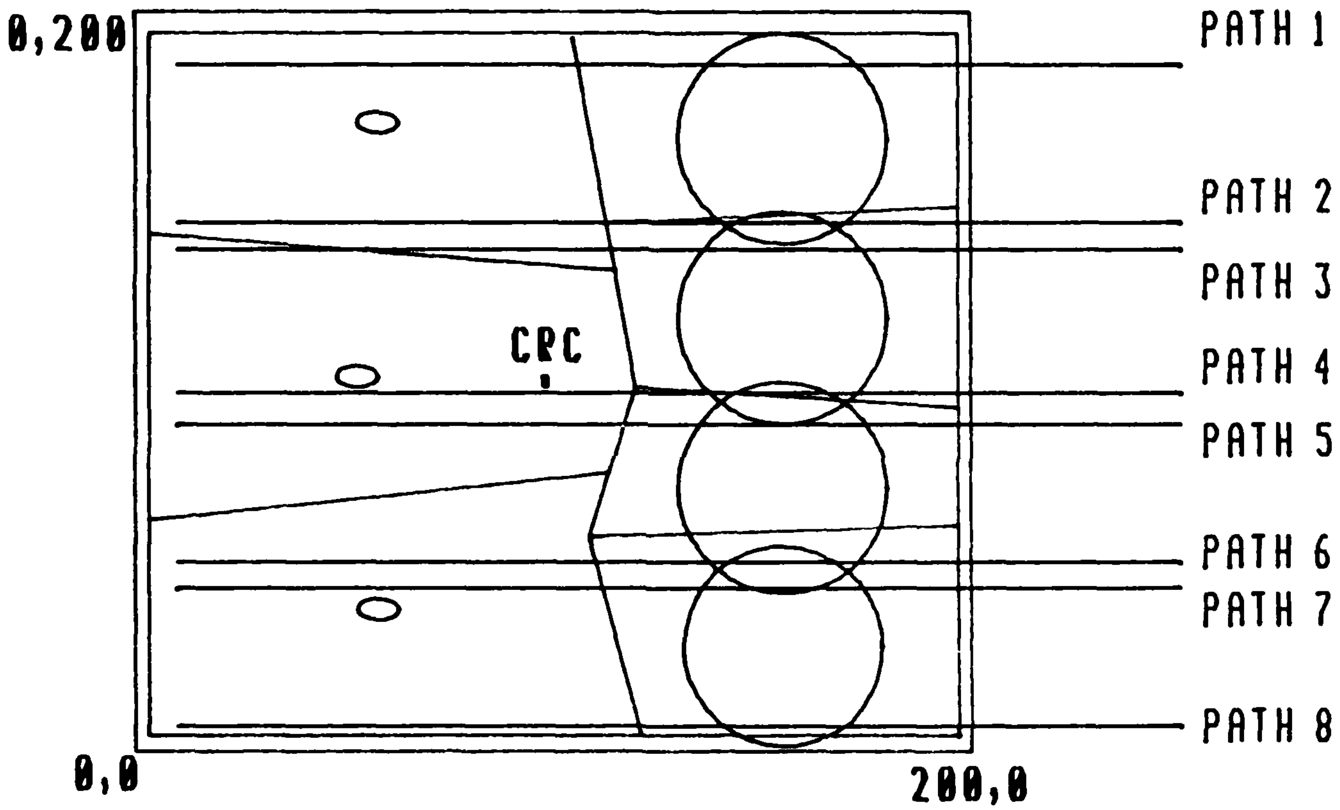


Figure 13. Thesis Air Defense Scenario.

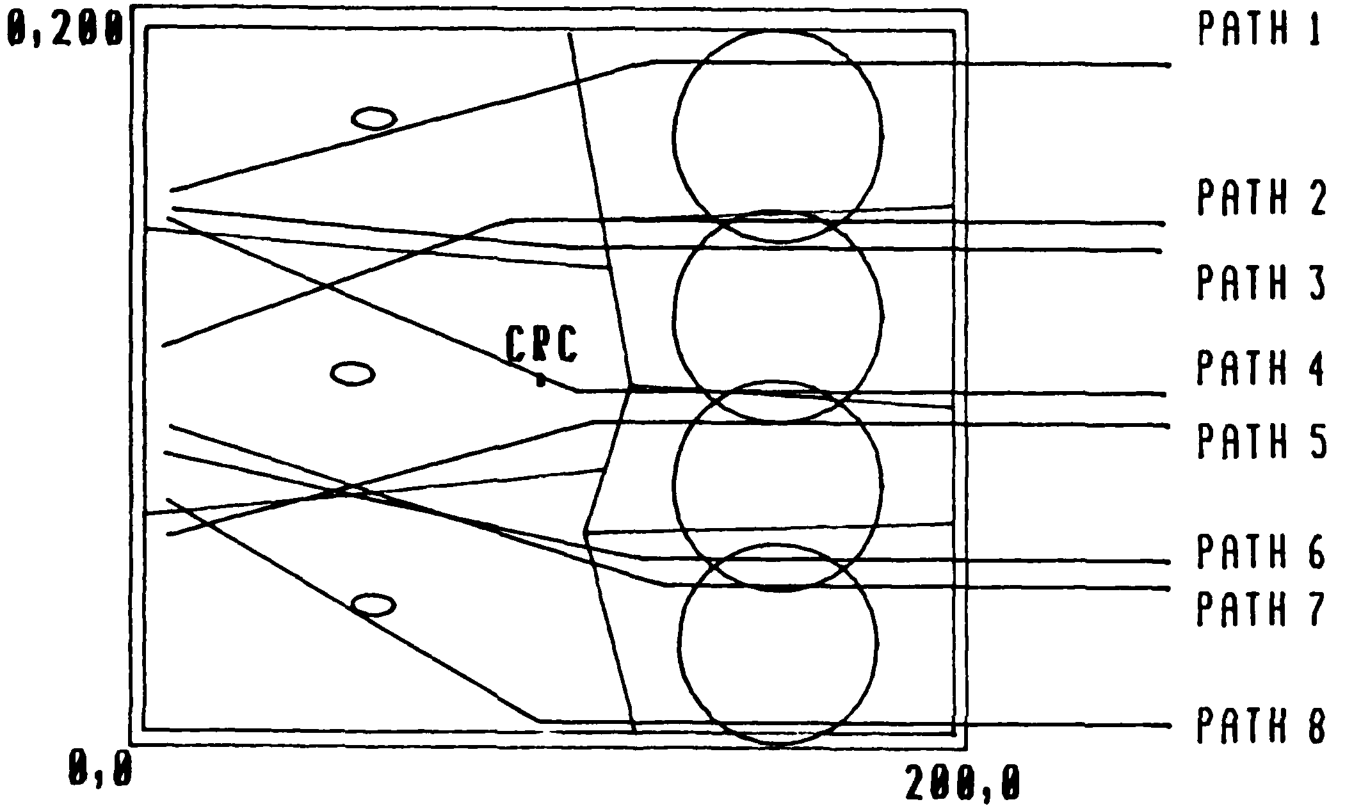


Figure 14. Modified Penetrator Path Scenario.

lists the statistical results for this comparison also. All of the above analyses essentially tested the model's sensitivity to the changes in the enemy's penetration tactics.

Sensitivity Analysis on Friendly Tactics

Several "friendly" tactics were also evaluated using the modified TADZ model. These tests are in addition to those already considered in the area of increased identification efficiency. These excursions include the command configuration of the surface to air missiles and a variation on the location of the CAP points for the air defense fighters.

SAM Command Configuration. The original scenario had the SAM's set up in a "square" configuration which permitted two platoons at each of the four SAM sites or batteries. This configuration allows for only two simultaneous intercepts at each site [Combined Arms Fundamentals, 1983:5.12,5.27]. The "triad" configuration puts one more platoon at each battery and therefore allows for three simul'aneous intercepts per site. While the actual number of missiles or platoons were not changed, the number of simultaneous intercepts was increased to three. Ten replications were run using the new configuration and the results were compared to the original scenario. The ANOVA indicated a difference in the means and the Duncan test confirmed that the triad configuration increased the number of hostiles that successfully penetrated the air defense zone. See Table XI. While examining the model and its outputs to understand the reason for the increase in the number of successful penetrators, a significant insight to the functioning and implications of the model was brought to light.

Table XI. Sensitivity on Friendly Tactics.

Scenario	No. Through (Mean)	F-stat Signif?	Duncan Test Signif?
Automated	38.0	----	----
SAM Config. (Triad)	49.6	Yes	Yes
CAP Forward 26 km	42.3	No	N/A
CAP Forward 13 km	41.3	No	N/A

Since the model output had been closely examined under the automated conditions, which were assumed to be the most stressful, it was felt that given longer processing times the model would continue to function as expected. However, when the TADZ network was re-examined, it was found that when the system sensed that a detection message would be backlogged by normal reporting (up the chain through the J&I section), the individual C^0s were allowed to engage the penetrators. This would only occur if such actions were the only method to prevent the penetrator from exiting the air defense zone without prosecution. Thus, as ID times were extended and larger backlogs occurred, more and more penetrators were attacked outright without the ID processing delay -- permitting fewer penetrators to escape. Thus the single identification processor providing a 10 second average processing time may be inadequate. Also, since the backlogged system can not execute as a total air defense system, the system begins to prosecute each penetrator as it enters each C^0s area of responsibility. In addition, since the system operates in the default mode, there is no dual prosecution resolution for overlapping areas of responsibility at the missile C^0s. Thus the SAM sites are only effective until the M&I section backlogs and then the additional server appears to cause increased conflict among the prosecutors allowing more penetrators to successfully pass through the air defense zone.

CAP Location. The base-line scenario had the CAP's located approximately in the center of their assigned areas or zones (see Figure 15). Moving these locations to a more forward location was hypothesized to improve the fighters' ability to engage and kill the incoming penetrators. The CAP locations were moved forward 26 kilometers (penetrator speed times the additional ID processing time). The CAP positions

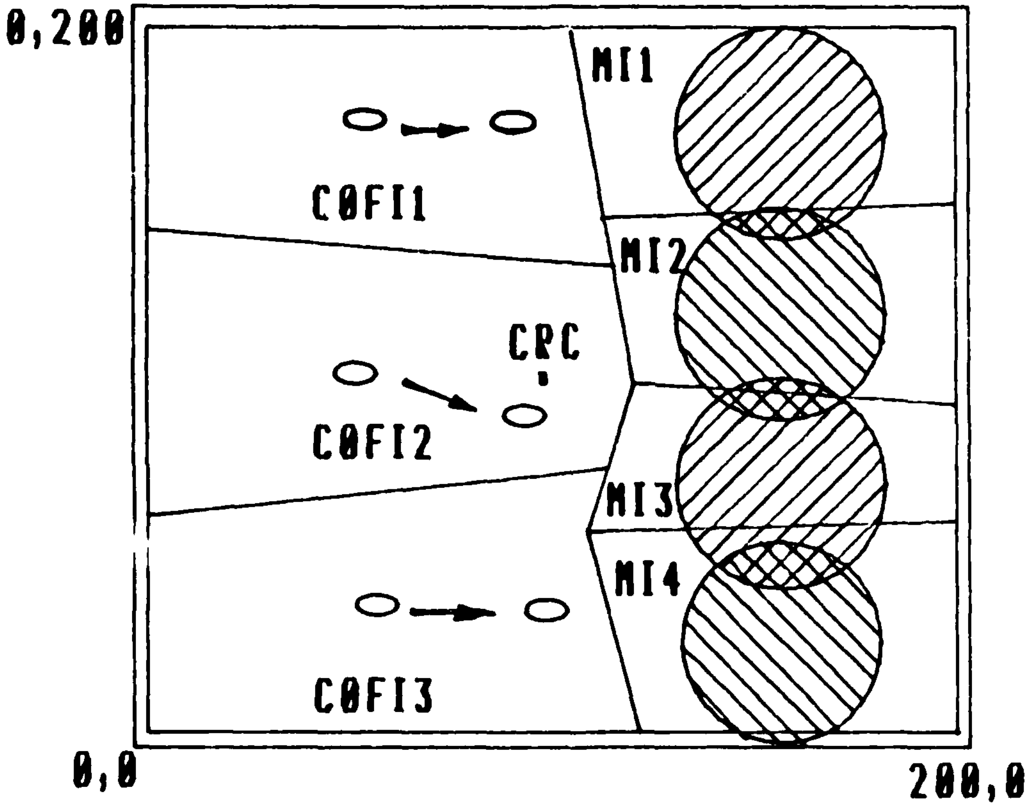


Figure 15. Forward CAP Locations.

were also tested at 13 kilometers forward (half the distance). The resulting average number of penetrators, for ten runs, through was not statistically different from the manual or automated system. These outputs are also included in Table XI.

The Research Question

Recall that the main purpose for this effort was to determine a range of optimal response rates to model the automated identification feature, specifically CIS-ISS. In addition, a comparison between the automated and manual systems was also proposed. The analysis thus far has evaluated an automated system based on the measures of accuracy, capability, and identification time. The question remains: does this projected automated system (CIS-ISS) do a better job than the current (manual) system? Table XII lists the factor levels to be used in this comparison.

Table XII. Comparison Factors.

Factor	CIS-ISS	Manual
Capability	2%	6%
Accuracy	99%	90%
ID Time	10 sec	112 sec

Using Wilcox's method, mentioned earlier in Chapter 2, the two systems were compared [Wilcox, 1985:45-54]. Using an initial 10 runs for the manual system, a mean of 35.6 hostile penetrators made it through the air defense zone. This is 12.36% of the total number of hostiles (35.6 / (320*.90)) available. It was hypothesized that if the CIS-ISS could reduce the percentage to 6% (half the manual) the system would be selected over the manual system. If the automated system was not as good as the manual system we would prefer the manual, and we would be indifferent for any performance between these two values. Specifically, select manual if the number through is greater than 36 (35.6) and select the automated if the number is less than 17 (17.28). If the number is between 17 and 36 the decision maker would be indifferent to the new system and would remain with the standard. Thus d^* = 9.5, and the midpoint of the indifference zone is 26.5. If the final automated mean minus the manual mean is less than d^* the test says to select the manual system, and if the difference is greater than d^* select the automated system.

Using the initial number of runs as 10, the mean of the automated system is 38 and the manual is 35.6 with sample variances of 4 and 7.305. The 'h' value for a probability of 95% correct decision is 2.61 [Wilcox, 1985:47]. The calculated number of samples needed is 10 and 10.21 for automated and manual respectively. The formulas used to calculate the number of samples are listed below. N_a is the number of runs required for the automated systems. N_m is the number of runs required for the manual system.

$$N_a = \max\ [10,\ (2.61/9.5)^2*(4)^2]$$

$$N_a = \max\ [10,\ 2.6]$$

$$N_m = \max\ [10,\ (2.61/9.5)^2*(7.805)^2]$$

$$N_m = \max\ [10,\ 10.21]$$

Therefore an additional sample was taken for the manual system and the new mean was 35.8. Since the difference was less than 9.5 (38 - 35.8 = 2.2), this test indicates that we should not select the automated system (as modeled).

Summary

This chapter has described the analysis of the results from the experimental design. While in some instances, the results were not as expected, they were understandable upon closer examination. The final analysis showed that a single ID processor as modeled is not adequate for identification purposes. Due to the backlog that occurs in both the automated and manual systems there were no significant operational differences between the two systems as modeled. In addition, the model sensitive to the enemy tactics employed while indifferent to the friendly tactics employed.

Chapter VI will contain a brief summary of the thesis, a restatement of the conclusions and include the authors' recommendations and suggestions for further study.

VI. Conclusions and Recommendations

Summary

In order to provide a framework for the conclusions and recommendations contained in this final chapter, a summary of the purpose, assumptions, scenario and the methodology will be presented.

Purpose. The goal of this thesis was to examine the range of working parameters required in the combat identification system - indirect subsystem (CIS-ISS) to prevent backlogs under simulated combat conditions in the control and reporting center (CRC). The model was analyzed to investigate the interactions between the different parameters used in the CIS-ISS.

Assumptions. Three major assumptions were used in modeling the CIS-ISS:

1. the identification sensor and supporting radars were collocated with approximately the same coverage.

2. the identification function would remain at the CRC movement and identification section (a man in the loop) with centralized inputs from all other radars and identification sensors.

3. the most stressful test of the CIS-ISS would be the employment of the TACS using a central European type scenario.

Scenario. The simulated test environment was a CRC using three CAP points using three squadrons of F-15s. The

CRC modeled controled 4 Hawk batteries with a total of 72 missiles ready to fire. The threat consisted of 320 penetrators flying in flights of 4 with 15 second spacing. The penetrators arrived along 8 separate paths. Ten percent of the penetrators were identified friendly and the remaining ones were hostile. See Figure 16 for a representation of the modeled air defense zone.

Methodology. Simulation was selected to test the parameters for the CIS-ISS. The simulation model used was the transient air defense zone (TADZ) model provided by Foreign Technology Division (FTD). It was modified to use identification codes, conduct identification intercepts and to selectively not prosecute penetrators based on the identification codes. Three parameters were used to model the CIS-ISS: capability (the percent of unknowns), accuracy (percent correct), and time to identify (how long the identification process takes). The three parameters were each tested at three factor levels and a 3^3 analysis of variance test was performed. Two measures of merit were used, total number of successful penetrators through the air defense zone and number of friendlies destroyed (fratricide).

The analysis of variance (ANOVA) showed that the significant factors for the first measure of merit, the number of successful penetrators, were accuracy and identification time. Capability was not significant at the

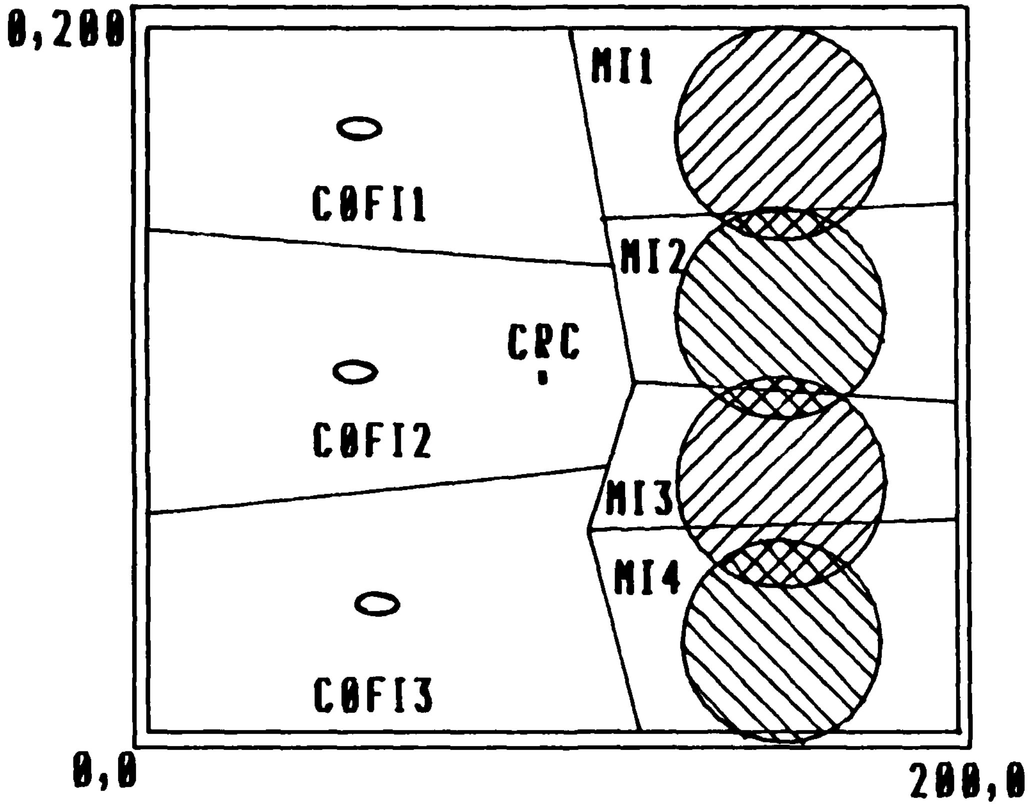


Figure 16. Thesis Air Defense Scenario.

levels tested due to the small number of friendly unknowns modeled in the scenario. Accuracy was a significant factor and as accuracy increased, the total number of successful penetrators and fratricides decreased. Identification time (ID) time was also significant. However, the effect was the opposite of what was anticipated. The number of successful penetrators decreased as ID time increased. ID time was not significant for fratricides.

Sensitivity tests conducted on variations in the penetration tactics were all significant indicating that the model is sensitive to changes in spacing between flights, size of the flights, and the paths flown. For this reason the conclusions presented are dependent on the scenario tested and should not be extrapolated beyond the threat tested in the model. Sensitivity tests on friendly tactics showed that the model was insensitive to changes in the combat air patrol (CAP) locations (forward positions) and sensitive to the number of simultaneous surface to air missile (SAM) intercepts. The greater the number of simultaneous intercepts allowed, the worse the system as a whole performed.

Conclusions

An analysis of the CIS-ISS/CRC as modeled led to the following two conclusions. First, the single centralized CIS-ISS processor desired by the operational users may not be feasible. Secondly, the use of "time to ID" as an operational parameter may not be an adequate measure. These two points will be discussed in the following paragraphs.

CIS-ISS Overload. The examination of model output showed that the reason for the opposite effect of ID time was because the identification processor was creating a backlog in message processing within the CRC. When the modeled CIS-ISS begins to backlog, rather than let penetrators through without processing, the CRC defaults to an 'autonomous mode' where each Hawk site and CAP point attempts to prosecute every penetrator within its assigned area of responsibility without awaiting an identification code. The longer the ID times, the sooner the system went into the default mode. The quicker the system enters the 'autonomous mode' the more penetrators are destroyed. In the autonomous mode, dual engagement of the same penetrator by two different SAMs or fighters may occur. Thus an increase in number of prosecutor servers causes poorer performance (more dual conflicts).

Even at the fastest processing time modeled for the CIS-ISS (10 seconds per ID) the system backlogs. The CIS-ISS, as modeled with a single central processor, can not handle the threat load presented. An simple queueing analysis of the arrival and service times also indicated the CIS-ISS would backlog.

A simplified queuing system of the thesis scenario can be visualized as shown in Figure 17. Several assumptions were made reduce the system to this simple queuing system. First,

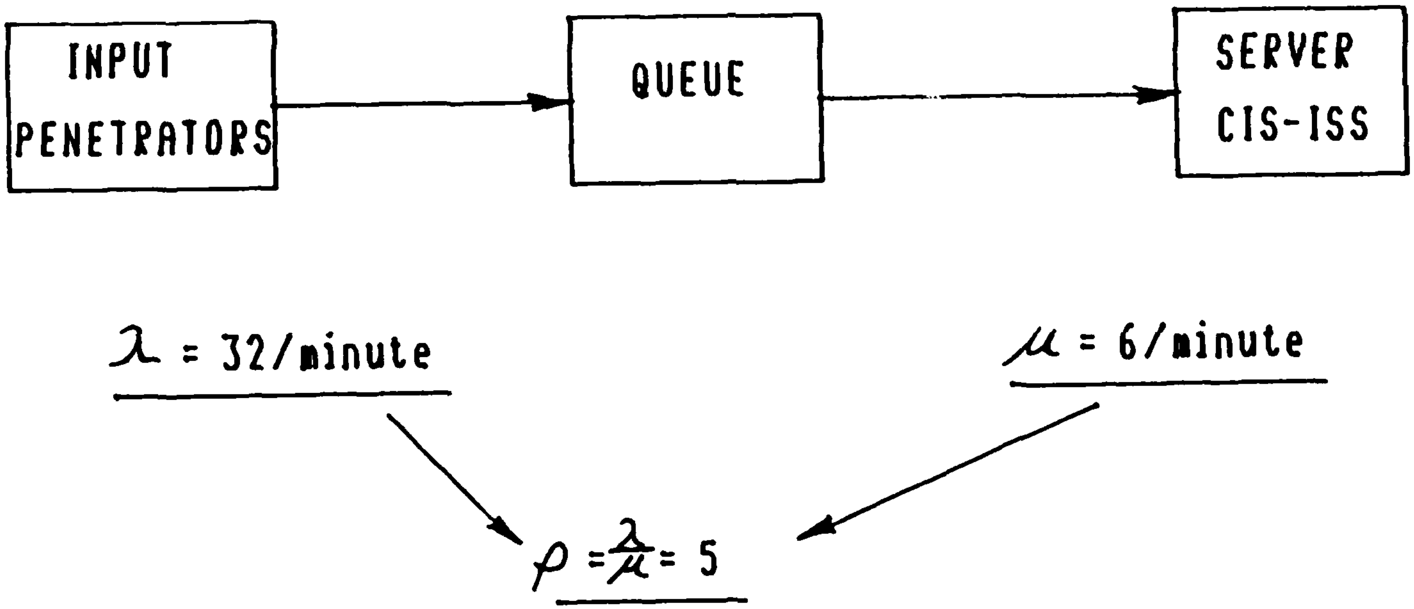


Figure 17. Simple Queuing Model.

the eight penetrator paths were assumed to present a single queue or customer population. Assuming that the CIS-ISS would identify the penetrator flights as a whole rather than as individual aircraft, the number of tracks or customers is reduced to 80 arrivals over a time span of approximately 2.5 minutes. This assumption yields an arrival rate, λ, of one penetrator every 2 seconds. Making the assumption that the CIS-ISS is a <u>single</u> server with a service time, μ, of one every 10 seconds, the utilization factor, ρ, for this queuing system would equal 5 (10 divided by 2). An intuitive conclusion at this point would indicate that this system cannot possibly work when the customers are arriving 5 times faster than they can be served. In analytical terms, such a queuing system cannot reach "steady state" unless the utilization factor is less than one [Hillier and Lieberman, 1980:417].

In order to follow through with some simple analytical calculations, several further adjustments would have to be made. Two solutions are immediately evident. The first would be to to insist on a CIS-ISS service time of much less than 10 seconds, a minimum value of 1.75 seconds would be required for the simplified system mentioned earlier. A two second response time identification response rate does not seem feasible, especially with the requirement for a man "in the loop." To verify that a one second response rate would case the model was run with a one second CIS-ISS response time. At the faster response rate the I&I section did not backlog. However, a backlog did occur in the weapons assignment section for the selection of air defense assets (SAMs). It should be noted that the time used to model the decision time for the SAMs (56 seconds) was based on the Eglin test data (a limited scenario) and may not be a valid response time for a higher threat environment.

A second approach to enable an increased CIS-ISS response rate would be to increase the number of servers to effectively decrease the average service time of the system. In the scenario described in this study, installing the CIS-ISS at each radar site would approach the required number of servers. This distributed CIS-ISS proposal would require the user to develop a change in the operational concept for a single centralized air space manager responsible for identification. The success of both of the previoulsy mentioned solutions, one second response time and multiple servers, are predicated on reducing the utilization factor to less than one.

The arrival rate in the used in the model has been dictated by what the authors believe to be a reasonable "worst case" scenario derived from the unclassified literature. Perhaps a simple study using the <u>actual</u> intelligence estimates of the threat would resolve the issue of whether or not the CIS-ISS can keep up with the expected load in an actual conflict. This simple queuing analysis, even though at a cursory level, does provide insight into this backlog problem. The utilization rate of the "system" as a whole must be reduced to increase the efficiency of the system. The central single processor for the CIS-ISS cannot "do the job" in the environment in which it will be expected to perform - as it was modeled in this thesis.

__ID Time as a Factor__. The use of identification processing speed as a CIS-ISS parameter may not be valid. It may not preserve the independence between tactical decisions to employ assets and the time to identify the penetrators. This same point was made by Major Hess (German Air Force [GAF], Brockzetel), a master controller--the equivalent of a WAO/Senior Director in the TACS. Major Hess suggested that

> in a combat situation, the amount of time (per se) needed to accomplish an ID was a less valid measure than the ability to have an unknown aircraft identified before it flew out of the effective engagement zone of a weapons system (e.g., surface-to-are missiles). The ingredients contained in the measure allude to the interrelationship between parameters inside/outside the system boundaries that must be considered when dealing with measures of system performance. Straight measures of timeliness/accuracy appear meaningless unless woven into the context of integrated air defense operations. [Preidis and Spaeth, 1985:3]"

__Recommendations__

__Distributed CIS-ISS__. Due to the saturation of the CIS-ISS as modeled, it is recommended that a further study be conducted to model a effect of a distributed identification function at __each__ CIS-ISS sensor verses the central processor desired by the operational users. The study should seek to analyze the benefits and trade-offs to distributing the identification process verses the current stated operational requirements for a centralized configuration.

Expanded Test Data. The distributions and times used in this study were based on limited sample data from a simple scenario employed in the May 1985 Eglin test. Additional analysis on the more expanded tests conducted in Europe should be used to test and validate the assumptions on the distributions used in this model.

Selective Prosecution. A study should be conducted using a selective prosecution based on identification of aircraft type and mission. For example, what benefits can be derived from prosecuting ground attack aircraft while not prosecuting hostile penetrating air-to-air fighters?

Review the TADZ/SLAM Model. During the process of modifying TADZ to implement the changes required, several problems were encountered in trying to compile and link the provided source code. In particular the original simulation language for alternative modeling (SLAM) source code provided could not be recompiled and linked without errors. While the compiled SLAM object file did execute satisfactorly with the modified TADZ object file, occasional unexplainable runtime errors did occur. Since the TADZ code did require complete recompilation to run on the VAX/VMS 11-785 (the code was developed on a VAX/VMS 11-780), it is not known whether the problem was due to the large system operating requirements (that approaches the system limits for a VAX 11-785), the differences between the computer systems or possible coding problems. This issue should be resolved to maintain the validity of the TADZ model and its documentation in future applications.

Bibliography

Aeronautical Systems Division, Air Force Systems Command. Contract F19628-83C-0054 with General Dynamics Corporation. Wright-Patterson AFB OH, 1983.

Albett, C. B. "C^3I Evolution Leaves a Lot to Chance," Defense Electronics Vol 16 No 1: 49-59 (January 1984).

Annino, Joseph S. and Edward C. Russell. "The Seven Most Frequent Causes of Simulation Analysis Failure - and How to Avoid Them," Interfaces Vol 11 No 3: 59-63 (June 1981).

Banks, Jerry and John S. Carson, II. Discrete-Event System Simulation. Englewood Cliffs NJ: Prentice Hall, Inc., 1984.

Battilega, John A. and Judith K. Grange (editors). The Military Applications of Modeling. Wright Patterson AFB OH: Air Force Institute of Technology Press, 1984.

Berg, Donald J. Catalog of Wargaming and Military Simulation Models. Washington D.C.: Studies, Analysis, and Gaming Agency, Organization of the Joint Chiefs of Staff, January 1980.

Bowman, Richard. Analyst, Command and Control Section. Personal interviews. Foreign Technology Division, Air Force Systems Command, Wright-Patterson AFB OH, 1 October 1985 to 1 May 1986.

Carson, John S., II and Averill M. Law. "Blocking and Confidence Intervals Using 'T' Statistics," Operations Research Vol 27 No 5: 1011-1025 (1979).

Chan, John W. "Tactical Air Control Simulator Correlates to Real World," National Defense Vol LXIX No 407: 20-27 (April 1985).

Cole, Lt. Col. Dennis L. A Conceptual Design for Modeling the Air War in Central Europe. Unpublished report. Air War College, Maxwell AFB AL, 1982 (AD-A118-917).

Coyne, James P. "Frontal Aviation's One-Two Punch," Air Force Magazine March 1985: 74,75.

CRC (Control and Reporting Center) Operational Test and Evaluation. OT&E No. 407L, 1972.

Department of the Army. U.S. Army Command and General Staff College, Department of Tactics. Combined Arms Fundamentals. Ft. Leavenworth, KS, 1983.

Department of Defense, Armed Forces Staff College. Joint Staff Officer's Guide. AFSC Pub 1. Norfolk, VA, July 1984.

Department of the Air Force, Headquarters Tactical Air Command. Tactical Air Control System (TACS), Surveillance and Control of Tactical Air Operations. TACR 55-44. Langley AFB, VA, 20 March 1975.

Donnelly, General Charles L., Jr., CINC USAFE. Project Warrior Address to AFIT Students. Air Force Institute of Technology (AU), Wright Patterson AFB, OH, 1 August 1985.

Fishman, George S. "Estimating Sample Size in Computer Simulation Experiments," Management Science Vol 18: 21-38 (September 1971).

"Friend or Foe - The New NATO IFF Standard," Defense Electronics Vol 15 No 3: 34 (March 1983).

Gardner, Col. Robert E. "Employment of Tactical Air Control Radars," Signal Vol 37 No 3: 55-58 (November 1982).

Gibbons J. D. et al. Selecting and Ordering Populations: A New Statistical Methodology. New York: John Wiley and Sons, 1977.

Greenburg, Capt. Abe. "An Outline of Wargaming," Naval War College Review Vol 34: 93-97.

Hausrath, Alfred K. Venture Simulation in War, Business and Polotics. New York: McGraw Hill, Inc., 1971.

Hicks, Charles R. Fundamental Concepts in the Design of Experiments. New York: Holt, Rinehart, and Winston, Inc., 1973.

Hillier, Frederick S. and Gerald J. Lieberman. Introduction to Operations Research. Oakland, CA: Holden-Day, Inc., 1980.

Hines, William W. and Douglas C. Montgomery. Probability and Statistics in Engineering and Management Science. New York: John Wiley and Sons, 1980.

HQ TAC/DRC. "Required Capability-Ground Electronic Support Measures (ESM) Sensor for Platform Identification." Electronic Message. 192030Z. August 1985.

Innis, George and Eric Rexstad. "Simulation Model Simplification Techniques," Simulation. Vol 41 No 1: 7-15 (July 1983).

Jones, Lt. Col. Paul M., Chief, Tactical C^2 Systems Division. Personal interview. Headquarters Tactical Air Command, Langley AFB VA, 23 December 1985.

Kirk, Roger E. Experimental Design: Procedures for the Behavioral Sciences. Belmont, CA: Wadsworth, Inc., 1982.

Kleijnen, Jack P. C. Statistical Techniques in Simulation, Part I and II. New York: Marcel Dekker, 1975a.

-----. "Two Variance Reduction Techniques," Management Science Vol 21: 1176-1185 (June 1975b).

Law, Averill M, David W. Kelton, Simulation Modeling and Analysis. New York: McGraw-Hill, 1982.

Merriman, Mark P. et al. C3 Systems Dynamics Project, Software Report, Transient Air Defense Zone (TADZ), User's Guide. Burlington, MA: Alphatech, Inc., 1984.

Merriman, Mark P. and John R. Dowdle. "A Comparison of Two Air Defense Models," Proceedings of 6th MIT/ONR Workshop on C^3 Systems. Cambridge, MA: Laboratory for Information and Decision Systems, MIT, 1983 (AD-A138 614).

Montgomery, Douglas C. Design and Analysis of Experiments. New York: John Wiley and Sons, 1984.

O'Brien, Lt. Col. Personal Interview. Combat Identification System Program Office, Wright Patterson AFB OH, 21 August 1985.

Perini, Michail B., Major USAF. "Telling Ours from Theirs", Air Force Magazine Vol 68 No 6: 80-83 (June 1985).

Powers, Capt Brian E. Tactical Air Defense in Central Europe. MA thesis. Naval Postgrduate School, Monterey, CA, June 1981 (AD-B089 934).

Preidis, Rosemarie J., *CRC System Model Using SLAM*. Unpublished paper. Air Force Human Resources Laboratory, Air Force Systems Command, Wright Patterson AFB, OH, July 1985a.

-----. Personal Interviews. Air Force Human Resources Laboratory, Wright Patterson AFB OH, 1 August through 1 January 1985b.

Preidis, Rosemarie J. and Major Raymond L. Spaeth. Trip Report No. TA30450. Air Force Human Resources Laboratory, Wright Patterson AFB OH, 7 November 1985.

Pritsker, A. Alan B. and Claude Dennis Pegden. *Introduction to Simulation and SLAM II*. New York: John Wiley and Sons, 1984.

Quade, E. S. and W. I. Boucher (editors). *Systems Analysis and Policy Planning: Applications in Defense*. New York: Elsevier, 1977.

Quattromani, Lt. Col. Anthony F. *Catalog of Wargaming and Military Simulation Models*. Organization of the Joint Chiefs of Staff. Studies, Analysis, and Gaming Agency. May 1982 (AD-A115 950).

Sargent, R.G. "Verification and Validation of Simulation Models," In *Progress in Modeling and Simulation* (F.E. Cellier, ed.). London: Academic Press, 1982.

Schlesinger, Stewart et al. "Terminology for Model Credibility," *Simulation* Vol 32 No 3: 103-104 (1979).

Shannon, Robert E. "Simulation: A Survey with Research Suggestions," *AIIE Transactions*, pp. 289-301 (September 1975).

Shindall, Dr. Joel. "ESM Sensor for Non-cooperative Target Classification," Unpublished Paper. San Jose CA: Watkins Johnson Company.

Shubik, Martin. *Games for Society, Business and War: Towards a Theory of Gaming*. New York: Elsevier, 1975.

Soviet Military Power. Washington, D.C.: U.S. Government Printing Office, 1985.

Spaeth, Major Raymond L. Personal Interviews. Air Force Human Resources Laboratory, Wright Patterson AFB OH, 1 September 1985 through 15 May 1986.

Stiling, Tom, Mitre-Bedford Corporation. Slides from CIS-ISS Demonstration Concept Briefing (CIS-ISS Test Plan Working Group), Eglin AFB, FL, 10 May 1984.

TAF-SON. Tactical Air Forces Statement of Operational Need No. 305-79, 1979.

Van Horn, Richard L. "Validation of Simulation Results," *Management Science* Vol 17 No 5 (January 1971).

Wilcox, Rand R. "On Comparing Treatment Effects to a Standard when the Variances are Unknown and Unequal." *Journal of Educational Statistics, Vol 10*: 45-54 (Spring 1985).

VITA

Major Robert C. MacFarlane was born on 31 January 1952 in Spokane, Washington. In May of 1974 he received a Bachelor of Arts degree in Mathematics from the Virginia Military Institute and was commissioned in the USAF. He served his first tour of duty at the 341st Strategic Missile Wing, Malmstrom AFB, Montana. While at Malmstrom, he served as a Wing Instructor and Chief of Proficiency training. He also had three assignments as an air weapons controller: as an instructor at the 72nd Tactical Control Flight, Fort Monroe, Virginia (a Forward Air Control Post); as the operations officer at Detachment 1, 621st Tactical Air Control Squadron, Yongmun San, Korea (a Control and Reporting Post); and as a staff officer at the Tactical Air Forces Interoperability Group (TAFIG), Headquarters Tactical Air Command, Langley AFB, Virginia. At Langley he served as Chief, Tactical System Branch and was instrumental in publishing the operations concept for the ground attack control center and producing the Pacific Air Forces Tactical Air Control System Command and Control Improvements Plan. He served at TAFIG until entering the Air Force Institute of Technology, in August 1984. Major MacFarlane also has an MBA from Golden Gate University, San Fransisco, California.

Permanent Address: 14797 Daley Lane
Woodbridge, VA 22193

VITA

Captain D. Michael McGuire was born on 26 March 1955 in Bellefonte, Pennsylvania. he graduated from high school in Towanda, Pennsylvania, in 1973 and attended the United States Air Force Academy from which he received a Bachelor of Science degree in Civil Engineering and direct commission in the USAF in 1977. After completing navigator training and F-111 upgrade training, he was assigned to the 494 Tactical Fighter Squadron, RAF Lakenheath, England in 1978. While in England, he served as an F-111 instructor weapons systems officer (IWSO), a standardization evaluation officer, and a scheduling officer. In 1981, Captain McGuire was reassigned to the 389 Tactical Fighter Training Squadron as an IWSO, and the assistant chief of scheduling. He served the 389th in these capacities until entering the School of Engineering, Air Force Institute of Technology, in August 1984.

Permanent address: R.D. 3, Box 45
Towanda, Pennsylvania 18848

APPENDIX A: TADZ CODE CHANGES

This appendix contains the changed FORTRAN subroutines that require modification and recompiled. The actual changes that are required in each file are preceeded by "C*** " and then comments to explain the code changes. Except for the changes to SERVTIME.FOR which can be found in the file USERF.FOR, all require the addition of the PENIDENT.INC in the common block definitions. The only change to USERF.FOR is to change the first service time from a uniform to an exponential distribution.

After the code changes have been made to the subroutines the individual routines need to be recompiled into the TADZ object libraries provided on the TADZ tape. This can be accomplished using the "RC subroutine.for" command. This command, and all other commands mentioned in this brief are provided in the "LOGIN.COM" file provided on the TADZ tape.

After all the changed subroutines have been recompiled, the entire object library needs to be 'relinked' to produce a new executable image. This new executable image can be produced using the "TLINK" command.

<u>PENIDENT.INC</u> (Page A-2) This file needs to be added to the TADZ.INCLUDE directory prior to the recompilation commands.

PENINT.FOR (Page A-4) The identification probabilities contained at the end of this file are cumulative probabilites. The actual probabilities for each code (recall Figure 9) can be found by taking the differences between the successive probabilities.

SHORTFIRE.FOR and LONGFIRE.FOR (Page A-8 and A-18) These code changes are explained in Chapter IV and are not logical and straight forward.

CCMATCH.FOR (Page A-30) It should be remembered that a change in the number of COFI will require recompilation of the code in this file. In addition the organization of the data files assumes that all fighter C^{O}s are listed first in the data execution file (CASE.DAT).

USERF.FOR (Page A-36) This file actually contains two separate subroutines, USERF and SERVICE_TIME. The second file, SERVICE_TIME is the routine requiring the change. The only change required is to change the FIRST_MESSAGE_TIME from a UNIFORM distribution to an EXPONENTIAL one. The orginal code has been commented out and is still in the file.

The actual files used are listed in the order just covered in the remainder of this Appendix. Each file starts on a separate page.

```
C
C     THIS COMMON BLOCK MUST FOLLOW AFTER 'PARAMETER.DIM'
C     AS THAT FILE CONTAINS THE DEFINITION OF THE VALUE FOR
C     MAX_NUM_PENETRATORS
C
C
      COMMON /PIDTAB/PEN_IDENT(MAX_NUM_PENETRATORS)
C
      REAL PEN_IDENT
C
C     PENETRATOR IDENTITY STATUS TABLE
C              CODE      ID              ACTUAL
C
C              1      UNKNOWN         HOSTILE
C              2      UNKNOWN         FRIENDLY
C              3      HOSTILE         HOSTILE
C              4      HOSTILE         FRIENDLY
C              5      FRIENDLY        HOSTILE
C              6      FRIENDLY        FRIENDLY
C              7      DELIBERATE NONE PROSECUTION
C              8      FRIENDLY ORIGINAL UNKNOWN
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     PENINT
C
C     'PENINT' INITIALIZES THE PENETRATOR TABLES FOR USE IN THE DYNAMIC
C     SEGMENT OF THE SIMULATION.
C
C     ALPHATECH,INC.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE PENINT
C
      IMPLICIT NONE
C
C     ***********************  INCLUDE COMMON BLOCKS  ***********************
C
      INCLUDE 'DCO:PARAMETER.DIM'
      INCLUDE 'DCO:SYSLIMITS.PRM'
      INCLUDE 'DCO:PENTIM.PRM'
      INCLUDE 'DCO:PENST.PRM'
      INCLUDE 'DCO:WRHEAD.PRM'
      INCLUDE 'DCO:PATHPEN.BLK'
      INCLUDE 'DCO:PENS.INC'
      INCLUDE 'DCO:PENTABLE.INC'
      INCLUDE 'DCO:EVENTS.INC'
C
C*** COMMON BLOCK ADDED FOR ID FEATURE
C
      INCLUDE 'DCO:PENIDENT.INC'
C
C     ******************** EXTERNAL FUNCTIONS REFERENCED ********************
C
      REAL ENTRY_TIME
C                 TIME A PENETRATOR ENTERS THE REGION OF RESPONSIBILITY OF
C                 A SYSTEM NODE.
C
C*** FUNCTION ADDED FOR ID FEATURE
C
      REAL UNFRM
C                 A RANDOM DRAW TO DETERMINE THE ID CODE
C
C     *********************  LOCAL VARIABLE DEFINITIONS  *********************
C
      INTEGER I
C                 DO LOOP INDEX
      INTEGER J
C                 DO LOOP INDEX
      INTEGER CURR_PATH
C                 PENETRATOR PATH OF CURRENT PENETRATOR
      INTEGER CURR_TYPE
C                 TYPE OF THAT PENETRATOR
```

```
      LOGICAL FOUND
C               HAS THAT TYPE OF PENETRATOR ALREADY BEEN SEEN ALONG THIS PATH?
C
C***  ADDED FOR ID FEATURE
C
      REAL    ID_RND_VAR
C               VARIABLE TO USE WITH ID CODE
C
C     ***************************  EXECUTABLE CODE  ******************************
C
C     FOR ALL PATHS, ZERO OUT THE NUMBER OF PENETRATOR TYPES
C
      DO I=1,NPATHSI
         N_TYPES_FOR_PATH(I)=0
         DO J=1, MAX_NUM_PEN_TYPES
               TYPES_FOR_PATH(I,J)=0
         ENDDO
      ENDDO
C
C     ZERO OUT THE PENETRATOR TABLE
C
      DO I = 1, MAX_NUM_PENETRATORS
         DO J = 1, NUM_PEN_TABLE_ENTRIES
               PEN_TABLE (I, J) = 0.0
         ENDDO
      ENDDO
C
C
C     INITIALIZE THE PENETRATOR TABLE, AND ESTABLISH THE BLOCK PATHPEN,
C     WHICH STORES ALL TYPES OF PENETRATORS THAT MAY COME DOWN A GIVEN PATH.
C
      DO I = 1, NUMPEN
C
         PEN_TABLE (I, EXISTENCE)       = TRUE
         PEN_TABLE (I, PENETRATOR_TYPE) = PEN_TYP (I)
         PEN_TABLE (I, PENETRATOR_PATH) = PEN_PATH (I)
         PEN_TABLE (I, SYS_ENTRY)       = ENTRY_TIME (I, CD, 1)
C
C     THE SYSTEM ENTRY TIME IS SET TO THE TIME OF THE FIRST EVENT ON
C     THE PENETRATOR'S EVENT CALENDAR, WHICH IS TYPICALLY ENTRY INTO
C     SOME RADAR COVERAGE ZONE.
C
         PEN_EVENT(I)=0
         PEN_WAYPOINT(I)=1
         PEN_TARGET(I)=0
         DO J=1,N_WH_TYPE
               PEN_WARHEAD(I,J)=PEN_BOMBS(PEN_TYP(I),J)
         ENDDO
C
C     INITIALIZE THE NUMBER OF CD'S AFTER THE PENETRATOR
C
         N_CD_AFTER_PEN(I)=0
```

```
C
C     POSSIBLY ADD TO THE LIST OF PENETRATOR TYPES ALONG A GIVEN PATH.
C
          CURR_PATH = PEN_PATH (I)
          CURR_TYPE = PEN_TYP  (I)
          FOUND = .FALSE.
          IF (N_TYPES_FOR_PATH (CURR_PATH) .GT. 0) THEN
                DO J = 1, N_TYPES_FOR_PATH (CURR_PATH)
                    IF (TYPES_FOR_PATH (CURR_PATH, J) .EQ.
     1                                        CURR_TYPE) THEN
                          FOUND = .TRUE.
                    ENDIF
                ENDDO
          ENDIF
          IF (.NOT. FOUND) THEN
                N_TYPES_FOR_PATH (CURR_PATH) =
     1                                        N_TYPES_FOR_PATH (CURR_PATH) + 1
                TYPES_FOR_PATH (CURR_PATH, N_TYPES_FOR_PATH
     1                                        (CURR_PATH)) = CURR_TYPE
          ENDIF
C
C     CLEAR ECM TABLES
C
          DO J = 1, MAX_JAMMED_SITES
              PEN_ECM (I, J) = 0
          ENDDO
C
      ENDDO
C
C***   CODE ADDED TO INITIALIZE THE PENETRATOR ID CODES
C
C
      DO I = 1, MAX_NUM_PENETRATORS
C
          ID_RND_VAR = UNFRM(0.,1.,5)
C
          IF (ID_RND_VAR .LE. 0.018) THEN
               PEN_IDENT(I) = 1
          ELSEIF (ID_RND_VAR .LE. 0.02) THEN
             PEN_IDENT(I) = 2
          ELSEIF (ID_RND_VAR .LE. 0.02098) THEN
             PEN_IDENT(I) = 3
          ELSEIF (ID_RND_VAR .LE. 0.89416) THEN
             PEN_IDENT(I) = 4
          ELSEIF (ID_RND_VAR .LE. 0.99118) THEN
             PEN_IDENT(I) = 5
          ELSEIF (ID_RND_VAR .LE. 1.0) THEN
             PEN_IDENT(I) = 6
          ENDIF
C
      ENDDO
C
```

```
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SHORT_FIRE
C
C     SUBROUTINE 'SHORT_FIRE' SIMULATES THE SHORT RANGE PORTION OF
C     THE MISSION.
C
C     ALPHATECH, INC.
C     AIR FORCE CONTRACT F33657-81-C-2150
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE SHORT_FIRE (TIME, PEN, COIP, PACKED_UCLASS, UITEM,
     &                     SUB_F_REQ,N_NEW_F_REQ,NEW_MODEL_REQ,
     &                     NEW_F_REQ,NEW_SUB_F_REQ,T_NEW_F_REQ)
C
      IMPLICIT NONE
C
C     ************************ INCLUDE COMMON BLOCKS ************************
C
      INCLUDE 'DCO:PARAMETER.DIM'
      INCLUDE 'DCO:EXECUTIVE.DIM'
      INCLUDE 'DCO:STATUSNAM.DIM'
      INCLUDE 'DCO:COINIT.PRM'
      INCLUDE 'DCO:PENSI.PRM'
      INCLUDE 'DCO:WEPARA.PRM'
      INCLUDE 'DCO:COUNIT.INC'
      INCLUDE 'DCO:PENS.INC'
C
C*** COMMON BLOCK ADDED FOR ID FEATURE
C
      INCLUDE 'DCO:PENIDENT.INC'
C
C     ********************** INPUT VARIABLE DEFINITIONS **********************
C
      REAL      TIME
C               CURRENT TIME
      INTEGER   PEN
C               PENETRATOR NUMBER
      INTEGER   COIP
C               CO INDEX
      INTEGER   PACKED_UCLASS
C               UNIT CLASS
      INTEGER   UITEM
C               UNIT ITEM INDEX
      INTEGER SUB_F_REQ
C               SUB FUNCTION REQUEST IDENTIFIER
      INTEGER N_NEW_F_REQ
C               NUMBER OF SUBSEQUENT FUNCTION REQUESTS
      INTEGER NEW_MODEL_REQ(MAX_INTERFACE_SIZE)
C               NEW MODEL REQUESTS
      INTEGER NEW_F_REQ(MAX_INTERFACE_SIZE)
```

```
C               THE NEW FUNCTION REQUESTS
      INTEGER NEW_SUB_F_REQ(MAX_INTERFACE_SIZE)
C               THE NEW SUBFUNCTION REQUESTS
      REAL      T_NEW_F_REQ(MAX_INTERFACE_SIZE)
C               THE SCHEDULING TIME OF THE NEW FUNCTION REQUEST
C
C     ********************** LOCAL VARIABLE DEFINITIONS **********************
C
      CHARACTER*4 REPORT
C               STATUS ON SHOTS TAKEN BY PROSECUTOR
      REAL      TNEXT
C               NEXT ITERATION TIME
      REAL      P_SPEED
C               PENETRATOR SPEED
      CHARACTER*4 STATUS
C               MISSION STATUS AT OUTPUT
      REAL      VCRUISE
C               CRUISE SPEED
      REAL      TURN
C               UNIT TURN RATE
      REAL      FUEL_TO_BASE
C               FUEL NEEDED TO GET TO BASE
      REAL      RESID_FUEL
C               RESIDUAL FUEL
      REAL      URN
C               UNIFORM RANDOM NUMBER
      REAL      T_ORIENT
C               ORIENTATION TIME
      REAL      RANGE
C               INTERCEPT RANGE
      REAL      ASPECT
C               ASPECT ANGLE
      REAL      VWH
C               WARHEAD SPEED
      REAL      PK
C               KILL PROBABILITY
      REAL      DIST
C               DISTANCE TO BASE
      REAL      JUNK_PVEL(3)
C               PENETRATOR VELOCITY -- NEVER USED HERE
      REAL      UPOS(3)
C               UNIT POSITION
      INTEGER   PTYPE
C               PENETRATOR TYPE
      INTEGER   UNPACKED_UCLASS
C               UNIT TYPE INDEX
      LOGICAL   PEN_FIRE
C               PENETRATOR FIRE FLAG
      LOGICAL WEAPON_FOUND
C               DID WARHEAD LOCATE A WEAPON?
      LOGICAL   NONE
C               WEAPONS USED FLAG
```

```
      LOGICAL   KILL
C               SUCCESSFUL KILL FLAG
      REAL      TEXERI
C               TIME OF PENETRATOR GEOGRAPHIC EXIT FROM THE REGION
      CHARACTER*(UNIT_CLASS_STRING_LENGTH) P_UCLASS_STRING
C               THE TEXT FORM OF UNIT CLASS
C
C     *****************  EXTERNAL FUNCTIONS REFERENCED  **********************
C
      REAL      UNFRM
C               RETURNS UNIFORM RANDOM VARIABLE
      REAL      PEN_KILL
C               RETURNS PENETRATOR KILL
      REAL      NEAREST_BASE
C               RETURNS NEAREST BASE DISTANCE
      INTEGER   P_TYPE
C               RETURNS PENETRATOR TYPE
      REAL      EXIT_TIME
C               THE EXIT FROM A REGION FUNCTION
D     LOGICAL TRACE_ON
C               TRACE ON FLAG
      CHARACTER*32 DESCRIBE_STATUS
C               CONVERTS INTERNAL STATUS INTO STRING FOR OUTPUT
      CHARACTER*(UNIT_CLASS_STRING_LENGTH) UNIT_CLASS_STRING
C               THE FUNCTION THAT ASSIGNS P_UCLASS_STRING
C
C     ************************ EXECUTABLE CODE ********************************
C
D     IF (TRACE_ON ()) THEN
D         WRITE (TRACEFILE. *) ' '
D         WRITE (TRACEFILE, *) 'INPUT VARIABLES FOR SUBROUTINE ',
D    +                                'SHORT_FIRE'
D         WRITE (TRACEFILE. *) 'TIME',TIME
D         WRITE (TRACEFILE. *) 'PENETRATOR'.PEN
D         WRITE (TRACEFILE. *) 'CURRENT CO',COIP
D           P_UCLASS_STRING = UNIT_CLASS_STRING(COIP.PACKED_UCLASS,
D    +                                                            PACKED)
D         WRITE (TRACEFILE, *) 'UNIT CLASS'. P_UCLASS_STRING,
D    +                                              'UNIT ITEM'.UITEM
D     ENDIF
C
      TEXERI=EXIT_TIME(PEN.CO.COIP)
      REPORT=NONE_STATUS
      UNPACKED_UCLASS = CO_UNIT_TYPES (COIP, PACKED_UCLASS)
C
C     SET PENETRATOR TYPE
C
      PTYP = P_TYPE(PEN)
C
C     SET TURN RATE
C
      TURN = TURN_RATE(UNPACKED_UCLASS)
```

```
C
C     SET SPEED
C
      VCRUISE = CRUISE_SPEED(UNPACKED_UCLASS)
C
C     SET POSITION
C
      CALL UNIT_KINEM(TIME,COIP,PACKED_UCLASS,UITEM,UPOS)
      DIST = NEAREST_BASE(COIP,UPOS)
      FUEL_TO_BASE = DIST/VCRUISE
      RESID_FUEL = (FUEL_TO_BASE + PI/TURN) *
     +                                CRUISE_TO_FIGHT (UNPACKED_UCLASS)
C
C     REPORT RESULTS SO FAR IF TRACE IS ACTIVATED
C
D     IF (TRACE_ON ()) THEN
D         WRITE (TRACEFILE, *) 'PENETRATOR TYPE',PTYP
D         WRITE (TRACEFILE, *) 'FI CRUISE SPEED', VCRUISE
D         WRITE (TRACEFILE, *) 'UNIT POSITION',UPOS
D         WRITE (TRACEFILE, *) 'DIST TO BASE',DIST
D         WRITE (TRACEFILE, *) 'FUEL TO BASE',FUEL_TO_BASE
D         WRITE (TRACEFILE, *) 'RESIDUAL FUEL',RESID_FUEL
D     ENDIF
C
C     SET REFERENCE
C
      CALL SET_REF(TIME,UPOS,COIP,PACKED_UCLASS,UITEM)
C
C***  BEFORE ALLOWING FOR ATTRITION DETERMINE IF UNKNWOWN
C***  INTERCEPTS ARE FRIENDLY OR HOSTILE
C***  CODE ADDED FOR IDENTIFICATION INTERCEPT
C
C***  IF UNKNOWN FRIENDLY THEN SET STATUS TO
C***  MISS_STATUS TO FREE CO AND CHANGE ID TO
C***  CODE (8) TO PREVENT FUTURE PROSECUTION
C
      IF (PEN_IDENT (PEN) .EQ. 2 ) THEN
          TNEXT = TIME
          STATUS = MISS_STATUS
          PEN_IDENT(PEN) = 8
          GO TO 200
      ENDIF
C
C***  END CODE FOR IDENTIFICATION INTERCEPT
C
C     ACCOUNT FOR ATTRITION
C
      URN = UNFRM(0.,1.,5)
      IF (URN .GT. 0.5) THEN
          PEN_FIRE = .TRUE.
      ELSE
          PEN_FIRE = .FALSE.
      ENDIF
C
C     PENETRATOR FIRES FIRST
C
      IF (PEN_FIRE) THEN
          PK = PEN_KILL(PTYP,UNPACKED_UCLASS)
          CALL SIM_ENG(PK,KILL)
D         IF (TRACE_ON ()) THEN
D               WRITE (TRACEFILE, *) 'PENETRATOR FIRING FIRST AT FI'
D               WRITE (TRACEFILE, *) 'PK',PK
D               WRITE (TRACEFILE, *) 'FI KILLED?', KILL
D         ENDIF
          IF (KILL) THEN
                CALL WEA_DEAD(TIME,COIP,PACKED_UCLASS,UITEM)
                TNEXT = TIME
                IF (UNIT_STAT (COIP, PACKED_UCLASS, UITEM)
     +                                        .EQ. UNIT_DEAD_STATUS) THEN
                    STATUS = UNIT_DEAD_STATUS
                    GO TO 200
                ENDIF
          ENDIF
      ENDIF
C
C     FI ATTACKS
C
      REPORT= FIRE_STATUS
      RANGE = 0
C
C     COMPUTE ASPECT ANGLE
C
      ASPECT = UNFRM(0.,PI,5)
C
C     GET WARHEAD AND SIMULATE ENGAGEMENT OUTCOME
C
      CALL WARHEAD (COIP, PACKED_UCLASS, UITEM, PTYP, RANGE, ASPECT,
     +                              .TRUE., VWH, PK, WEAPON_FOUND, NONE)
C
      IF (.NOT. WEAPON_FOUND) THEN
          WRITE (ERROR_FILE, *) 'ERROR IN SHORT_FIRE'
          WRITE (ERROR_FILE, *) 'WARHEAD DID NOT LOCATE A WEAPON'
          CALL UERR (FATAL)
      ENDIF
C
C     IF OUT OF WEAPONS, THEN RETURN TO BASE
C
      IF(NONE) THEN
           UNIT_STAT(COIP,PACKED_UCLASS,UITEM)= BASE_STATUS
      ENDIF
C
      CALL SIM_ENG(PK,KILL)
D     IF (TRACE_ON ()) THEN
D         WRITE (TRACEFILE, *) 'FI ATTACKS'
```

```
D          WRITE (TRACEFILE, *) 'WARHEAD SPEED', VWH
D          WRITE (TRACEFILE, *) 'SALVO PK',PK
D          WRITE (TRACEFILE, *) 'PENETRATOR KILLED?', KILL
D          WRITE (TRACEFILE, *) 'WEAPONS DEPLETED', NONE
D      ENDIF
C
C      PENETRATOR DESTROYED
C
       IF (KILL) THEN
C
          TNEXT = TIME
          STATUS = DEAD_STATUS
          GO TO 200
C
C      IF MISSED CHECK FUEL AND UPDATE DESTINATION
C
       ELSE IF (FUEL_R(COIP,PACKED_UCLASS,UITEM) .GE. RESID_FUEL) THEN
C
C         IF NO WEAPONS SEND TO BASE
C
          IF (NONE) THEN
D              IF (TRACE_ON ()) THEN
D                 WRITE (TRACEFILE, *) 'FI OUT OF WEAPONS, ',
D    +                                            'RETURN TO BASE'
D              ENDIF
               TNEXT = TIME
               STATUS = BASE_STATUS
               UNIT_STAT(COIP,PACKED_UCLASS,UITEM)= BASE_STATUS
               GO TO 200
          ENDIF
C
C         CHECK PENETRATOR SPEED
C
          P_SPEED = PEN_SPEED(PEN_TYP(PEN))
C
C         INFEASIBLE
C
          IF (P_SPEED .GT. DASH_SPEED(UNPACKED_UCLASS)) THEN
               TNEXT = TIME
               STATUS = MISS_STATUS
               GO TO 200
          ENDIF
C
C         FEASIBLE FOR REENGAGEMENT
C
          STATUS = SHORT_STATUS
C
C         COMPUTE ORIENTATION TIME
C
          URN = UNFRM(0..PI.5)
          T_ORIENT = URN/TURN
C
```

```
C          SET UNIT AT PENETRATOR
C
           TNEXT = TIME + T_ORIENT
           CALL PEN_KINEM(TNEXT,PEN,UPOS,JUNK_PVEL)
C
C          UPDATE DESTINATION
C
           CALL SET_DES(TNEXT,UPOS,COIP,PACKED_UCLASS,UITEM)
C
C      NOT ENOUGH FUEL TO REENGAGE
C
       ELSE
C
C          SET UP THE CONTROLLER RELEASE AND RETURN TO BASE
C
           TNEXT = TIME
           UNIT_STAT(COIP,PACKED_UCLASS,UITEM)= BASE_STATUS
           STATUS = BASE_STATUS
           GO TO 200
       ENDIF
C
C      PENETRATOR FIRES SECOND
C
       IF (.NOT. PEN_FIRE) THEN
           PK = PEN_KILL(PTYP,PACKED_UCLASS)
           CALL SIM_ENG(PK,KILL)
D          IF (TRACE_ON ()) THEN
D                WRITE (TRACEFILE, *) 'PENETRATOR FIRING SECOND AT FI'
D                WRITE (TRACEFILE, *) 'PK',PK
D                WRITE (TRACEFILE, *) 'KILL OF FI BY PENETRATOR',KILL
D          ENDIF
           IF (KILL) THEN
                 CALL WEA_DEAD(TIME,COIP,PACKED_UCLASS,UITEM)
                 TNEXT = TIME
                 IF (UNIT_STAT (COIP,PACKED_UCLASS,UITEM) .EQ.
     +                                           UNIT_DEAD_STATUS) THEN
                     STATUS = UNIT_DEAD_STATUS
                     GO TO 200
                 ENDIF
           ENDIF
       ENDIF
200    CONTINUE
C
D      IF (TRACE_ON ()) THEN
D          WRITE (TRACEFILE, *) 'FINAL STATUS IN SUBROUTINE ',
D    +                                'SHORT_FIRE: ', DESCRIBE_STATUS (STATUS)
D      ENDIF
C
C      CANCEL ALL MISSIONS FOR ALL UNITS ASSIGNED TO A PENETRATOR
C
       IF(STATUS.EQ. DEAD_STATUS) THEN
C
```

```
C         FREE_CO_AFTER_PEN WILL DISENGAGE ALL UNITS IN ALL CO'S
C         PURSUING THIS PENETRATOR
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=PRIMITIVE_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=FREE_CO_AFTER_PEN_MODEL
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C         NOTIFY THE NETWORK OF PENETRATOR CANCELLATION
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=TERM_PEN_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=0
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C     COLLECT DESTRUCTION STATISTICS
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=STATISTICS_ENTRY_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=TERM_STATS_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=0
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C     DISENGAGE UNIT FOR MISSES
C
      ELSE IF(STATUS.EQ. MISS_STATUS) THEN
C
C         FREE THIS CO ONLY FROM PURSUIT
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=PRIMITIVE_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=FREE_ONE_CO_AFTER_PEN_MODEL
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C         HANDOVER FOR ESCAPES -- TEXERI = 0 IFF THE PENETRATOR HAS
C         GOTTEN OUT OF REACH OF THIS CO
C
C

          IF (PEN_IDENT(PEN) .EQ. 8) THEN
                TNEXT = TEXERI
          ENDIF
C
          IF(TEXERI.GT.0.0) THEN
              N_NEW_F_REQ=N_NEW_F_REQ+1
              NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
              NEW_F_REQ(N_NEW_F_REQ)=LOAD_QUEUE_MODEL
              NEW_SUB_F_REQ(N_NEW_F_REQ)=0
              T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
          ELSE
```

```
              N_NEW_F_REQ=N_NEW_F_REQ+1
              NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
              NEW_F_REQ(N_NEW_F_REQ)=HAND_OVER_MODEL
              NEW_SUB_F_REQ(N_NEW_F_REQ)=0
              T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
          ENDIF
C
C     FREE SERVER NOW
C
      ELSE IF(STATUS.EQ. UNIT_DEAD_STATUS) THEN
C
C         FREE THIS CO ONLY
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=PRIMITIVE_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=FREE_ONE_CO_AFTER_PEN_MODEL
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C         LOAD QUEUE MODEL FOR DEAD UNITS
C
          IF(TEXERI.GT.0.0) THEN
              N_NEW_F_REQ=N_NEW_F_REQ+1
              NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
              NEW_F_REQ(N_NEW_F_REQ)=LOAD_QUEUE_MODEL
              NEW_SUB_F_REQ(N_NEW_F_REQ)=0
              T_NEW_F_REQ(N_NEW_F_REQ)=TIME
          ELSE
              N_NEW_F_REQ=N_NEW_F_REQ+1
              NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
              NEW_F_REQ(N_NEW_F_REQ)=HAND_OVER_MODEL
              NEW_SUB_F_REQ(N_NEW_F_REQ)=0
              T_NEW_F_REQ(N_NEW_F_REQ)=TIME
          ENDIF
C
C     FREE SERVER AND RETURN UNIT TO BASE
C
      ELSE IF(STATUS.EQ. BASE_STATUS) THEN
C
C         FREE THIS CO
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=PRIMITIVE_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=FREE_ONE_CO_AFTER_PEN_MODEL
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C         LOAD THE QUEUE FOR A RETRY FOR PROSECUTOR TERMINATED
C
          IF(TEXERI.GT.0.0) THEN
              N_NEW_F_REQ=N_NEW_F_REQ+1
              NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
              NEW_F_REQ(N_NEW_F_REQ)=LOAD_QUEUE_MODEL
              NEW_SUB_F_REQ(N_NEW_F_REQ)=0
              T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
          ELSE
              N_NEW_F_REQ=N_NEW_F_REQ+1
              NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
              NEW_F_REQ(N_NEW_F_REQ)=HAND_OVER_MODEL
              NEW_SUB_F_REQ(N_NEW_F_REQ)=0
              T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
          ENDIF
C
C     OTHERWISE THE STATUS IS SHORT_STATUS
C
      ELSE
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=CO_ENG_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=SHORT_FIRE_MODEL
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
      ENDIF
C
      IF(REPORT.EQ. FIRE_STATUS) THEN
C
C         SCHEDULE ENGAGEMENT STATISTICS COLLECTION ROUTINES.
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=STATISTICS_ENTRY_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=FIRE_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=0
          T_NEW_F_REQ(N_NEW_F_REQ)=TIME
      ENDIF
C
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     LONG_FIRE
C
C     SUBROUTINE 'LONG_FIRE' CONDUCTS THE LONG RANGE FIRING.
C
C     ALPHATECH, INC.
C     AIR FORCE CONTRACT F33657-81-C-2150
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE LONG_FIRE(TIME,PEN,COIP,PACKED_UCLASS,UITEM, SERVER,
     &                 SUB_F_REQ,N_NEW_F_REQ,NEW_MODEL_REQ,
     &                 NEW_F_REQ,NEW_SUB_F_REQ,T_NEW_F_REQ)
C
      IMPLICIT NONE
C
C     ************************ INCLUDE COMMON BLOCKS ************************
C
      INCLUDE 'DCO:PARAMETER.DIM'
      INCLUDE 'DCO:EXECUTIVE.DIM'
      INCLUDE 'DCO:STATUSNAM.DIM'
      INCLUDE 'DCO:COINIT.PRM'
      INCLUDE 'DCO:WEPARA.PRM'
      INCLUDE 'DCO:FIRERANGE.BLK'
      INCLUDE 'DCO:COSERV.INC'
      INCLUDE 'DCO:COUNIT.INC'
C
C***  COMMON BLOCK ADDED FOR ID FEATURE
C
      INCLUDE 'DCO:PENIDENT.INC'
C
C     ********************* INPUT VARIABLE DEFINITIONS **********************
C
      REAL      TIME
C               CURRENT TIME
      INTEGER   PEN
C               PENETRATOR NUMBER
      INTEGER   COIP
C               CO INDEX
      INTEGER   PACKED_UCLASS
C               UNIT TYPE INDEX
      INTEGER   UITEM
C               UNIT ITEM INDEX
      INTEGER SERVER
C               SERVER CONDUCTING THIS ENGAGEMENT
      INTEGER SUB_F_REQ
C               SUB FUNCTION REQUEST IDENTIFIER
      INTEGER N_NEW_F_REQ
C               NUMBER OF SUBSEQUENT FUNCTION REQUESTS
      INTEGER NEW_MODEL_REQ(MAX_INTERFACE_SIZE)
C               NEW MODEL REQUESTS
```

```
      INTEGER NEW_F_REQ(MAX_INTERFACE_SIZE)
C                 THE NEW FUNCTION REQUESTS
      INTEGER NEW_SUB_F_REQ(MAX_INTERFACE_SIZE)
C                 THE NEW SUBFUNCTION REQUESTS
      REAL        T_NEW_F_REQ(MAX_INTERFACE_SIZE)
C                 THE SCHEDULING TIME OF THE NEW FUNCTION REQUEST
C
C     ********************** LOCAL VARIABLE DEFINITIONS **********************
C
      LOGICAL SUCF
C                 SUCCESS FLAG ON WEAPONS SELECTION
      REAL        TNEXT
C                 NEXT ITERATION TIME
      CHARACTER*4 STATUS
C                 STATUS VARIABLE FOR RETURN DIRECTION
      REAL        RAD_RAN_TIME
C                 TIME PENETRATOR IN RADAR RANGE
      REAL        VUNIT
C                 UNIT SPEED
      REAL        CON_RAT
C                 UNIT FUEL CONSUMPTION RATE
      REAL        VCRUISE
C                 UNIT CRUISE SPEED
      REAL        RR
C                 AUTONOMY RANGE
      REAL        TPRIME
C                 TEMPORARY TIME VARIABLE
      REAL        R_R_T0
C                 RADAR RANGE TIME VARIABLE
      REAL        R_R_T1
C                 RADAR RANGE TIME VARIABLE
      REAL        TEX
C                 EXIT TIME
      REAL        PP
C                 DISTANCE SQUARED
      REAL        TINT
C                 INTERCEPT TIME
      REAL        DIST
C                 DISTANCE
      REAL        FUEL_NEED
C                 NEEDED FUEL
      REAL        T_FUEL
C                 REMAINING FUEL
      REAL        RANGE
C                 INTERCEPT RANGE
      REAL        VMAG
C                 VELOCITY MAGNITUDE
      REAL        ASPECT
C                 ASPECT ANGLE
      REAL        VWH
C                 WARHEAD SPEED
      REAL        PK
```

```
C               KILL PROBABILITY
      REAL      R_MAX
C               MAXIMUM FIRING RANGE
      LOGICAL WEAPON_FOUND
C               DID WARHEAD FIND A WEAPON TO USE?
      REAL      TWH_INT
C               WARHEAD INTERCEPT TIME
      REAL      TEMP
C               TEMPORARY VARIABLE
      REAL      PPOS(3)
C               PENETRATOR POSITION
      REAL      PVEL(3)
C               PENETRATOR VELOCITY
      REAL      UPOS(3)
C               UNIT POSITION
      REAL      UVEL(3)
C               UNIT VELOCITY
      REAL      DELP(3)
C               RELATIVE POSITION
      REAL      EST_INT_POS(3)
C               ESTIMATED INTERCEPT POSITION
      REAL      DEL_POS(3)
C               RELATIVE POSITION TO INTERCEPT
      INTEGER   UNPACKED_UCLASS
C               UNIT TYPE INDEX
      INTEGER   I
C               DO LOOP INDEX
      INTEGER   PTYP
C               PENETRATOR TYPE
      LOGICAL   FLAG
C               CONTINUE FLAG
      LOGICAL   NONE
C               WEAPON EXHAUSTION FLAG
      LOGICAL   KILL
C               KILL FLAG
      CHARACTER*4 REPORT
C               REPORT STATUS ON SHOTS AGAINS A PENETRATOR
C
C     ************************* FUNCTION DEFINITIONS *************************
C
      REAL      EXIT_TIME
C               RETURNS EXIT TIME
      REAL      INNER_PRODUCT
C               RETURNS INNER PRODUCT
      REAL      NEAREST_BASE
C               RETURNS DISTANCE TO NEAREST BASE
      REAL      NEXT_WAY_POINT
C               RETURNS NEXT WAYPOINT TIME
      INTEGER   P_TYPE
C               RETURNS PENETRATOR TYPE
      CHARACTER*32 DESCRIBE_STATUS
C
```

```
      CHARACTER*(UNIT_CLASS_STRING_LENGTH) P_UCLASS_STRING
C               TEXT FORM OF UNIT CLASS
C
C     ******************** EXETERNAL FUNCTIONS REFERENCED *******************
C
C               CONVERTS INTERNAL STATUS INTO STRING FOR OUTPUT
D     LOGICAL TRACE_ON
C               TRACE ON FLAG
      CHARACTER*(UNIT_CLASS_STRING_LENGTH) UNIT_CLASS_STRING
C               FUNCTION THAT SETS THE VALUE P_UCLASS_STRING
C
C     **************************** EXECUTABLE CODE **************************
C
C     GET INDEX
C
D     IF (TRACE_ON ()) THEN
D        WRITE (TRACEFILE, *) ' '
D        WRITE (TRACEFILE, *) 'LONG_FIRE'
D        WRITE (TRACEFILE, *) 'TIME',TIME
D        WRITE (TRACEFILE, *) 'PENETRATOR',PEN
D        P_UCLASS_STRING = UNIT_CLASS_STRING (COIP,PACKED_UCLASS,
D    +                                                                PACKED
D        WRITE (TRACEFILE, *) 'UNIT CLASS' ,P_UCLASS_STRING,
D    +                                    ' UNIT ITEM ',UITEM
D     ENDIF
      REPORT= NONE_STATUS
      UNPACKED_UCLASS = CO_UNIT_TYPES (COIP, PACKED_UCLASS)
C
C     SET SPEEDS, CONSUMPTION RATES
C
      VUNIT = DASH_SPEED(UNPACKED_UCLASS)
      VCRUISE = CRUISE_SPEED(UNPACKED_UCLASS)
      CCN_RAT = CRUISE_TO_DASH(UNPACKED_UCLASS)
C
C     SET AUTONOMY RANGE
C
      RR = AUTONOMY_RANGE(COIP,PACKED_UCLASS,UITEM)
C
C     SET REFERENCE LOCATION
C
      CALL UNIT_KINEM(TIME,COIP,PACKED_UCLASS,UITEM,UPOS)
      CALL SET_REF(TIME,UPOS,COIP,PACKED_UCLASS,UITEM)
C
C     COMPUTE RANGE
C
      CALL PEN_KINEM(TIME,PEN,PPOS,PVEL)
D     IF (TRACE_ON ()) THEN
D        WRITE (TRACEFILE, *) 'AUTONOMY RANGE', RR
D        WRITE (TRACEFILE, *) 'UNIT POSITION ', UPOS
D        WRITE (TRACEFILE, *) 'PEN  POSITION ', PPOS
D        WRITE (TRACEFILE, *) 'PEN  VELOCITY ', PVEL
D     ENDIF
```

```
      CALL VSUB(PPOS,UPOS,DELP)
      PP = INNER_PRODUCT(DELP,DELP)
      RANGE = SQRT(PP)
D     IF (TRACE_ON ()) THEN
D         WRITE (TRACEFILE, *) 'CURRENT RANGE ', RANGE
D     ENDIF
      IF (RANGE .LT. SHORT_RANGE) THEN
          TNEXT = TIME
          STATUS = SHORT_STATUS
          GO TO 200
      ENDIF
C
C     COMPUTE UNIT INTERCEPT TIME
C
      CALL INTERCEPT_TIME(TIME,PPOS,UPOS,PVEL,VUNIT,TINT,FLAG)
C
C     CHECK FLAG TO DETERMINE IF INTERCEPT IS PHYSICALLY POSSIBLE
C
      IF (.NOT. FLAG) THEN
          TNEXT = TIME
          STATUS = MISS_STATUS
          GO TO 200
      ENDIF
C
C     NOW, CAN INTERCEPT OCCUR BEFORE THE PENETRATOR LEAVES THIS CO?
C
      TEX = EXIT_TIME(PEN,CO,COIP)
D     IF (TRACE_ON ()) THEN
D         WRITE (TRACEFILE, *) 'EXIT TIME', TEX
D     ENDIF
      IF (TEX .LT. TINT) THEN
          TNEXT = TIME
          STATUS = MISS_STATUS
          GO TO 200
      ENDIF
C
C     CHECK FUEL FEASIBILITY
C
      DO I = 1,3
C
C         SET ESTIMATED INTERCEPT POSITION
C
          EST_INT_POS(I) = PPOS(I) + PVEL(I)*(TINT - TIME)
C
C         COMPUTE RELATIVE POSITION
C
          DEL_POS(I) = EST_INT_POS(I) - UPOS(I)
      ENDDO
D     IF (TRACE_ON ()) THEN
D         WRITE (TRACEFILE, *) 'ESTIMATED INTERCEPT POINT',
D    +                                   EST_INT_POS
D     ENDIF
```

```
C
C     COMPUTE REQUIRED FUEL
C
      PP = INNER_PRODUCT(DEL_POS,DEL_POS)
      DIST = NEAREST_BASE(COIP,EST_INT_POS)
      FUEL_NEED = CON_RAT*SQRT(PP)/VUNIT + DIST/VCRUISE
      T_FUEL = FUEL_R(COIP,PACKED_UCLASS,UITEM)
C
C     CHECK FEASIBILITY
C
      IF (T_FUEL .LT. FUEL_NEED) THEN
         TNEXT = TIME
         UNIT_STAT(COIP,PACKED_UCLASS,UITEM)= BASE_STATUS
         STATUS = MISS_STATUS
         GO TO 200
      ENDIF
C
C     SET DESTINATION
C
      CALL SET_DES(TINT,EST_INT_POS,COIP,PACKED_UCLASS,UITEM)
C
C     SET UNIT VELOCITY
C
      DO I = 1,3
         IF (TINT .NE. TIME) THEN
            UVEL(I) = DEL_POS(I)/(TINT - TIME)
         ENDIF
      ENDDO
C
C     CHECK WAYPOINTS
C
      TPRIME = NEXT_WAY_POINT(TIME,PEN)
C
C     SET RADAR RANGE TIME
C
      IF (PP .NE. 0) THEN
         R_R_T0 = TIME + (1 - RR/SQRT(PP))*(TINT - TIME)
      ENDIF
      R_R_T1 = MAX (R_R_T0, TIME)
C
C     MODIFY RADAR RANGE TIME BASED ON WAYPOINTS
C
      IF (R_R_T1 .LE. TPRIME) THEN
         RAD_RAN_TIME = R_R_T1
C
C        BASED ON CONTROL POLICY, FREE THE SERVER
C
         IF(CONTROL(COIP,PACKED_UCLASS,UITEM).EQ. LOOSE_STATUS) THEN
               IF(CO_SERVER_PEN (COIP, SERVER).EQ.PEN) THEN
                  STATUS= FREE_STATUS
                  TNEXT=RAD_RAN_TIME
                  GO TO 200
```

```
                 ENDIF
             ENDIF
          ELSE
             RAD_RAN_TIME = LARGE_REAL
          ENDIF
C
C         COMPUTE ASPECT ANGLE
C
          VMAG = SQRT(INNER_PRODUCT(PVEL,PVEL))
          IF (VMAG .NE. 0) THEN
             TEMP = INNER_PRODUCT(UVEL,PVEL)/(VUNIT*VMAG)
             IF (ABS(TEMP) .LT. 1.0) THEN
                   ASPECT = ACOS(TEMP)
             ELSE
                   ASPECT = ACOS(TEMP/ABS(TEMP))
             ENDIF
          ELSE
             ASPECT = 0
          ENDIF
C
C         GET FIRE RANGE
C
          CALL FIRE_RANGE(COIP,PACKED_UCLASS,UITEM,ASPECT,R_MAX,VWH)
C
C         EXTRAPOLATE THE PENETRATOR POSITION TO WARHEAD IMPACT TIME
C
          DO I=1,3
             DELP(I)=DELP(I)+PVEL(I)*R_MAX/VWH
          ENDDO
          PP=INNER_PRODUCT(DELP,DELP)
          PP = SQRT(PP) - PROSEC_DELTA
          IF (PP .GT. R_MAX) THEN
             CALL FIRE_TIME(TIME,PPOS,UPOS,PVEL,VWH,R_MAX,TINT,TNEXT,SUCF)
             IF(SUCF) THEN
                 STATUS = LONG_STATUS
             ELSE
                   STATUS= SHORT_STATUS
             ENDIF
             GO TO 200
          ENDIF
C
C***      BEFORE FIREING AT A DISTANCE. ADD CODE TO SIMULATE
C***      IDENTIFICATION INTERCEPT.  FORCE TO SHORTRANGE IF
C***      ID IS UNKNOWN
C
          IF (PEN_IDENT(PEN) .LT. 3) THEN
             CALL FIRE_TIME (TIME, PPOS, UPOS, PVEL, VWH, SHORT_RANGE,
         +                                 TINT, TNEXT, SUCF)
             STATUS = SHORT_STATUS
             GO TO 200
          ENDIF
C
```

```
C***  END ADDED CODE FOR ID
C
C     SIMULATE FIRING AT A DISTANCE
C
      PTYP = P_TYPE(PEN)
      CALL WARHEAD (COIP, PACKED_UCLASS, UITEM, PTYP, PP, ASPECT,
     +                              .FALSE., VWH, PK, WEAPON_FOUND, NONE)
D     IF (TRACE_ON ()) THEN
D        WRITE (TRACEFILE, *) 'RETURNS FROM WARHEAD'
D        WRITE (TRACEFILE, *) ' WEAPON FOUND = ', WEAPON_FOUND
D        IF (WEAPON_FOUND) THEN
D              WRITE (TRACEFILE, *) ' WARHEAD VELOCITY = ', VWH
D              WRITE (TRACEFILE, *) ' PK = ', PK
D              WRITE (TRACEFILE, *) ' WEAPON EXHAUSTION = ', NONE
D        ENDIF
D     ENDIF
C
C     IF NOT SUCCESSFUL, TRY SHORT FIRE
C
      IF (.NOT. WEAPON_FOUND) THEN
         CALL FIRE_TIME (TIME, PPOS, UPOS, PVEL, VWH, SHORT_RANGE,
     +                              TINT, TNEXT, SUCF)
C
C     FIRE_TIME DETERMINES WHEN A SHORT FIRE CAN OCCUR (AT TIME TNEXT)
C     BASED ON THE SHORT FIRING RANGE SHORT_FIRE.
C
         STATUS = SHORT_STATUS
         GOTO 200
      ENDIF
C
C     IF OUT OF WEAPONS, SIGNAL RETURN TO BASE
C
      IF(NONE) THEN
         UNIT_STAT(COIP,PACKED_UCLASS,UITEM)= BASE_STATUS
      ENDIF
C
C     COMPUTE WARHEAD INTERCEPT TIME
C
      CALL INTERCEPT_TIME(TIME,PPOS,UPOS,PVEL,VWH,TWH_INT,FLAG)
D     IF (TRACE_ON ()) THEN
D        WRITE (TRACEFILE, *) 'RETURNS FROM INTERCEPT_TIME'
D        WRITE (TRACEFILE, *) ' WARHEAD INTERCEPT TIME ', TWH_INT
D        WRITE (TRACEFILE, *) ' SUCCESSFUL INTERCEPT ',FLAG
D     ENDIF
C
C     CHECK FLAG
C
      IF (.NOT. FLAG) THEN
C
C        SET DESTINATION
C
         TNEXT = TINT
```

```
            STATUS = SHORT_STATUS
            GO TO 200
        ENDIF
        REPORT= FIRE_STATUS
C
C       GET UNIT POSITION AT IMPACT TIME
C
        CALL UNIT_KINEM(TWH_INT,COIP,PACKED_UCLASS,UITEM,UPOS)
C
C       SIMULATE KILL
C
        CALL SIM_ENG(PK,KILL)
D       IF (TRACE_ON ()) THEN
D           WRITE (TRACEFILE, *) 'RETURNS FROM SIM_ENG'
D           WRITE (TRACEFILE, *) ' EFFECTIVE PK = ', PK
D           WRITE (TRACEFILE, *) ' SUCCESSFUL KILL ', KILL
D       ENDIF
C
        IF (KILL) THEN
C
C           UPDATE PENETRATOR STATUS
C
            STATUS = DEAD_STATUS
            TNEXT=TWH_INT
        ELSE IF (NONE) THEN
            STATUS = BASE_STATUS
            TNEXT = TWH_INT
        ELSE
            STATUS = LONG_STATUS
            TNEXT = TWH_INT
        ENDIF
200     CONTINUE
C
C
D       IF (TRACE_ON ()) THEN
D           WRITE (TRACEFILE, *) 'FINAL STATUS IN SUBROUTINE ',
D     +                               'LONGFIRE: ', DESCRIBE_STATUS (STATUS)
D       ENDIF
C
C       FREE THE SERVER AND RETURN HERE
C
        IF(STATUS.EQ. FREE_STATUS) THEN
C
C           FREE SERVER MODEL
C
            N_NEW_F_REQ=N_NEW_F_REQ+1
            NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
            NEW_F_REQ(N_NEW_F_REQ)=PRIMITIVE_MODEL
            NEW_SUB_F_REQ(N_NEW_F_REQ)=FREE_SERVER_MODEL
            T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C           RETURN TO LONG FIRE AT RADAR RANGE TIME
```

```
C
         N_NEW_F_REQ=N_NEW_F_REQ+1
         NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
         NEW_F_REQ(N_NEW_F_REQ)=CO_ENG_MODEL
         NEW_SUB_F_REQ(N_NEW_F_REQ)=LONG_FIRE_MODEL
         T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C     SEND FI TO BASE
C
      ELSE IF(STATUS.EQ. BASE_STATUS) THEN
C
C        FREE THIS CO AFTER THE PENETRATOR
C
         N_NEW_F_REQ=N_NEW_F_REQ+1
         NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
         NEW_F_REQ(N_NEW_F_REQ)=PRIMITIVE_MODEL
         NEW_SUB_F_REQ(N_NEW_F_REQ)=FREE_ONE_CO_AFTER_PEN_MODEL
         T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C        LOAD THE QUEUE FOR BASE RETURNS
C
         N_NEW_F_REQ=N_NEW_F_REQ+1
         NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
         NEW_F_REQ(N_NEW_F_REQ)=LOAD_QUEUE_MODEL
         NEW_SUB_F_REQ(N_NEW_F_REQ)=0
         T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C     DISENGAGE UNIT FOR MISSES
C
      ELSE IF(STATUS.EQ.MISS_STATUS) THEN
C
C        FREE THIS CO AFTER THE PENETRATOR
C
         N_NEW_F_REQ=N_NEW_F_REQ+1
         NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
         NEW_F_REQ(N_NEW_F_REQ)=PRIMITIVE_MODEL
         NEW_SUB_F_REQ(N_NEW_F_REQ)=FREE_ONE_CO_AFTER_PEN_MODEL
         T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C        HANDOVER FOR ESCAPES
C
         N_NEW_F_REQ=N_NEW_F_REQ+1
         NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
         NEW_F_REQ(N_NEW_F_REQ)=LOAD_QUEUE_MODEL
         NEW_SUB_F_REQ(N_NEW_F_REQ)=0
         T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C     SEQUENCE THE LONG FIRE MODEL
C
      ELSE IF(STATUS.EQ.LONG_STATUS) THEN
         N_NEW_F_REQ=N_NEW_F_REQ+1
         NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
```

```
          NEW_F_REQ(N_NEW_F_REQ)=CO_ENG_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=LONG_FIRE_MODEL
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C     FREE SERVERS AND SEQUENCE THE SHORT FIRE MODEL
C
      ELSE IF(STATUS.EQ.SHORT_STATUS) THEN
C
C         SEQUENCE THE SHORT FIRE MODEL
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=PRIMITIVE_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=FREE_SERVER_MODEL
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C         SHORT FIRE MODEL
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=CO_ENG_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=SHORT_FIRE_MODEL
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C     CANCEL ALL MISSIONS AFTER THIS PENETRATOR
C
      ELSE IF(STATUS.EQ.DEAD_STATUS) THEN
C
C         FREE_CO_AFTER_PEN WILL DISENGAGE ALL UNITS IN ALL CO'S AFTER PEN
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=PROSECUTE_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=PRIMITIVE_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=FREE_CO_AFTER_PEN_MODEL
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C         NOTIFY THE NETWORK OF PENETRATOR TERMINATION
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=NETWORK_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=TERM_PEN_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=0
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
C     COLLECT DESTRUCTION STATISTICS
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=STATISTICS_ENTRY_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=TERM_STATS_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=0
          T_NEW_F_REQ(N_NEW_F_REQ)=TNEXT
C
```

```
      ENDIF
C
      IF(REPORT.EQ.FIRE_STATUS) THEN
C
C         SCHEDULE ENGAGEMENT STATISTICS COLLECTION ROUTINES.
C
          N_NEW_F_REQ=N_NEW_F_REQ+1
          NEW_MODEL_REQ(N_NEW_F_REQ)=STATISTICS_ENTRY_MODEL
          NEW_F_REQ(N_NEW_F_REQ)=FIRE_MODEL
          NEW_SUB_F_REQ(N_NEW_F_REQ)=0
          T_NEW_F_REQ(N_NEW_F_REQ)=TIME
      ENDIF
C
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CO_MATCH IS CALLED FROM THE CO INPUT STATISTICS EVENT NODE TO
C     DETERMINE WHETHER AN INCOMING MESSAGE CAN BE PROCESSED. THIS
C     REQUIRES THAT BOTH A DETECTION AND AN ASSIGNMENT BE LOCATED AT
C     THIS NODE, WITH DATABASE NUMBERS THAT HAVE BEEN RESOLVED TO
C     CORRESPOND TO THE SAME PENETRATOR, AND THAT THE CORRESPONDING
C     PENETRATOR HAS NOT YET LEFT THE CO'S COVERAGE REGION. SINCE
C     THERE ARE OTHER SORTS OF MESSAGES COMING INTO CO THAT DO NOT
C     REQUIRE MERGER, THIS ROUTINE MUST CHECK IF THE INCOMING MESSAGE
C     IS AN ENGAGEMENT TYPE OF MESSAGE, OR ONE OF THE OTHER TYPES--
C     REQUEST FOR SERVICE FROM A MINI-CO, OR A RESOURCE ALLOCATION
C     COMMAND.
C
C     ALPHATECH, INC.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE CO_MATCH
C
      IMPLICIT NONE
C
C*****COMMON BLOCKS AND PARAMETERS
C
      INCLUDE 'DCO:SLAMPARAM.DIM'
      INCLUDE 'DCO:SCOM1.SLM'
      INCLUDE 'DCO:PARAMETER.DIM'
      INCLUDE 'DCO:EXECUTIVE.DIM'
C
C    ***COMMON BLOCK ADDED FOR IDENTIFICATION FEATURE***
C
      INCLUDE 'DCO:PENIDENT.INC'
C
C*****EXTERNAL FUNCTIONS
C
      INTEGER GET_AWAIT_FILE
C                RETURNS SLAM INDEX FOR THE PARTICULAR AWAIT FILE NEEDED
      REAL EXIT_TIME
C                RETURNS DEPARTURE TIME FROM A GIVEN NODE--RETURNS 0 IF
C                THE PENETRATOR HAS ALREADY LEFT THE REGION, WHICH IS
C                WHAT IS TESTED FOR HERE
C     LOGICAL TRACE_ON
C                RETURNS TRUE IFF THE SLAM TRACE FACILITY IS ENABLED
C
C*****LOCAL VARIABLES
C
      INTEGER ASSIGN_FILE, DETECT_FILE
C                TEMPORARY VARIABLES TO STORE SLAM QUEUE NUMBERS FOR THE
C                TWO FILES IN WHICH MESSAGES WAIT TO BE MATCHED
      LOGICAL TEMP_ALERT, TEMP_DETECT
C                TEMPORARY VARIABLES THAT STORE THE STATE OF ALERT AND
C                DETECTION IN THE MESSAGE STORED IN ARRAY ATRIB (SCOM1)
```

```
      REAL TEMP_ATRIB (NUM_ATTRIBUTES)
C               STORES MESSAGES RETURNED FROM SEARCH_AWAIT_FILE UNTIL
C               INFORMATION FROM THEM MAY BE PROCESSED
      REAL RADAR_DETECTED
C               TEMPORARY USED IN TRANSFERRING INFORMATION FROM
C               TEMP_ATRIB TO ATRIB (OR VICE VERSA)
      INTEGER I
C               LOOP COUNTER
      INTEGER CLASS, ITEM
C               LOCATION OF CURRENT SYSTEM CLASS; DERIVED FROM ATRIB ARRAY
      INTEGER PEN
C               PENETRATOR TAIL NUMBER: ALSO OBTAINED FROM ATRIB
      INTEGER MESSAGE_TYPE
C               STORES INCOMING MESSAGE TYPE
      LOGICAL JUNK_BOOLEAN
      REAL    JUNK_ATRIB (NUM_ATTRIBUTES)
C               PARAMETERS TO CLEAR_FILE; NOT USED HERE
C
C*****EXECUTABLE CODE
C
C     LOCAL VARIABLES DERIVED FROM THE CONTENTS OF THE MESSAGE
C
      CLASS = NINT (ATRIB (CURRENT_CLASS))
      ITEM  = NINT (ATRIB (CURRENT_ITEM))
      PEN   = NINT (ATRIB (TRUE_PEN_NO))
      MESSAGE_TYPE = NINT (ATRIB (CO_MESSAGE_TYPE))
      TEMP_ALERT  = (ATRIB (ALERT_STATUS)     .EQ. TRUE)
      TEMP_DETECT = (ATRIB (CO_DETECTION_RPT) .EQ. TRUE)
      ASSIGN_FILE = GET_AWAIT_FILE (CLASS, ITEM, ASSIGN_CODE)
      DETECT_FILE = GET_AWAIT_FILE (CLASS, ITEM, DETECT_CODE)
      RADAR_DETECTED = ATRIB (RADAR_NUMBER)
C
      IF (ATRIB (HANDOVER_MESSAGE) .EQ. TRUE) THEN
          WRITE (ERROR_FILE, *) 'ERROR IN CO_MATCH'
          WRITE (ERROR_FILE, *)'MESSAGE RECEIVED WITH HANDOVER = TRUE'
          CALL UERR (FATAL)
C
C     IF THIS IS AN ENGAGEMENT MESSAGE, TAKE ACTION BASED ON THE
C     VALUES OF ALERT AND DETECT
C
      ELSEIF (MESSAGE_TYPE .EQ. CO_ENG_MODEL) THEN
          IF ((.NOT. TEMP_ALERT) .AND. (.NOT. TEMP_DETECT)) THEN
                WRITE (ERROR_FILE, *) 'ERROR IN CO_MATCH: ',
     1                                          'MESSAGE RECEIVED'
                WRITE (ERROR_FILE, *) 'WITH NEITHER ALERT OR DETECT ',
     2                                          'STATUS'
                CALL UERR (FATAL)
          ELSEIF (TEMP_ALERT .AND. (.NOT. TEMP_DETECT)) THEN
D               IF (TRACE_ON ()) THEN
D                   WRITE (TRACEFILE, *) 'COMATCH SEARCHING FOR ',
D    1                                                    'DETECTION REPORT'
D                   WRITE (TRACEFILE, *) 'FILE = ', DETECT_FILE
```

```
D              ENDIF
               IF (EXIT_TIME (PEN, CLASS, ITEM) .EQ. 0.0) THEN
                   CALL SEND_HANDOVER_MESSAGE (CLASS, ITEM, ATRIB)
                   II = DISCARD_CODE
               ELSE
                   II = ASSIGN_CODE
                   CALL SEARCH_AWAIT_FILE (DETECT_FILE, II,
     1                                                ATRIB, TEMP_ATRIB)
                   IF (II .EQ. MATCH_CODE) THEN
                         ATRIB (RADAR_NUMBER) = TEMP_ATRIB (RADAR_NUMBER)
                   ENDIF
               ENDIF
          ELSEIF ((.NOT. TEMP_ALERT) .AND. TEMP_DETECT) THEN
D              IF (TRACE_ON ()) THEN
D                  WRITE (TRACEFILE, *) 'COMATCH SEARCHING FOR ',
D    1                                                'ASSIGNMENT REPORT'
D                  WRITE (TRACEFILE, *) 'FILE = ', ASSIGN_FILE
D              ENDIF
               IF (EXIT_TIME (PEN, CLASS, ITEM) .EQ. 0.0) THEN
                   II = DISCARD_CODE
               ELSE
                   II = DETECT_CODE
                   CALL SEARCH_AWAIT_FILE (ASSIGN_FILE, II,
     1                                                ATRIB, TEMP_ATRIB)
C
C     TEMP_ATRIB RETURNS THE ASSIGNMENT MESSAGE--IF A MATCH HAS BEEN
C     FOUND, THEN THAT MESSAGE AND NOT THE DETECTION MESSAGE SHOULD BE
C     KEPT.
C
                   IF (II .EQ. MATCH_CODE) THEN
                         DO I = 1, NUM_ATRIB_USED
                             ATRIB (I) = TEMP_ATRIB (I)
                         ENDDO
                         ATRIB (RADAR_NUMBER)      = RADAR_DETECTED
                   ELSEIF (II .EQ. DETECT_CODE) THEN
C
C     IF NO ASSIGNMENT MATCH WAS FOUND, THEN CLEAR ALL PREVIOUS
C     DETECTION MESSAGES ABOUT THIS PENETRATOR FROM THIS RADAR OUT OF
C     THE DETECTION QUEUE AT THIS POINT, AND REPLACE THEM WITH THIS
C     MESSAGE. THE MOTIVATION FOR THIS IS THE FOLLOWING:
C        IT HAD BEEN IMPLEMENTED PREVIOUSLY THAT EACH RADAR CAUSED
C     ONLY ONE SIGHTING MESSAGE TO BE SENT TO ITS COMMANDING CO, UNDER
C     THE ASSUMPTION THAT THAT MESSAGE WOULD SUFFICE TO PRODUCE ANY
C     REASONABLE ASSIGNMENTS. THIS NEGLECTED, HOWEVER, THE DYNAMIC
C     NATURE OF THE DATABASE IN THE SYSTEM. IN PARTICULAR, UNDER THE
C     OLD SCHEME, IT COULD HAPPEN THAT A SIGHTING REPORT AND A
C     DETECTION REPORT ON A PENETRATOR WERE SENT TO THE SAME CO WITH
C     DIFFERENT DATABASE NUMBERS, SINCE THE ORIGINAL MESSAGES WERE
C     GENERATED AT A TIME WHERE THE C3 SYSTEM HAD NOT YET DETERMINED
C     THAT THE TWO PUTATIVE PENETRATORS WERE IN FACT ONE AND THE SAME.
C     IF THE MERGER OCCURRED WHEN THESE TWO MESSAGES WERE ALREADY AT
C     THE CO, THOSE MESSAGES COULD NOT BE RECONSIDERED. THIS LED TO
```

```
C     THE NEED TO CONTINUE TO SEND MESSAGES TO THE CO FROM THE SAME
C     RADAR ON A REGULAR BASIS, FOR THE NEW SIGHTING MESSAGE COULD BE
C     USED TO EFFECT A PROSECUTION IF IT TRANSPIRED THAT MERGER OF
C     DATABASE INFORMATION HAS OCCURRED IN THE INTERVAL BETWEEN THE
C     ARRIVAL OF THE LAST REPORT AND THE ARRIVAL OF THE CURRENT REPORT.
C     LEAVING ALL THESE MESSAGES AROUND, THOUGH, WOULD LEAD TO A VERY
C     SUBSTANTIAL SOFTWARE LOAD THAT IS IRRELEVANT TO MODEL EXECUTION,
C     FOR ONLY THE LATEST SIGHTING REPORT IS RELEVANT. HENCE THESE
C     MESSAGES ARE PURGED HERE BEFORE THE NEW MESSAGE IS FILED IN THE
C     FILE.
C        AS MENTIONED ABOVE, THIS REPAIR OCCURS AT A LATE DATE IN THE
C     DEVELOPMENT OF TADZ I. PROBABLY THE CLEANEST METHOD OF MODIFYING
C     THIS WOULD BE TO REVAMP THE MESSAGE ROUTING IMPOSED BY THE TESTS
C     DONE AT CO ON THE II VARIABLE, WHICH CURRENTLY CONTROLS THE
C     ROUTING TO THE APPROPRIATE QUEUE OF THE MESSAGE LEAVING THIS
C     NODE. BUT THAT WOULD REQUIRE A LOT OF CODE CHANGES. WHAT WILL BE
C     DONE INSTEAD IS TO SIMPLY REMOVE ALL MESSAGES FROM THE DETECTION
C     QUEUE AT THIS POINT, AND LET THE REFILING OF THE NEW MESSAGE BE
C     TAKEN CARE OF VIA THE OLD MECHANISM USING THE II VARIABLE.
C
                          JUNK_BOOLEAN = .TRUE.
                          DOWHILE (JUNK_BOOLEAN)
                              CALL CLEAR_ONE_DETECTION
     1                                         (DETECT_FILE, PEN, NINT(RADAR_DETECTED),
     2                                              JUNK_BOOLEAN, JUNK_ATRIB)
C
C     CLEAR_ONE_DETECTION FINDS ANY DETECTION OF THAT PENETRATOR BY
C     THAT RADAR THAT IS STORED IN THAT FILE, RETURNING TRUE IN THE
C     FOURTH ARGUMENT AND ITS ATTRIBUTES IN THE FIFTH ARGUMENT IF ONE
C     IS ACTUALLY FOUND.
C
                          ENDDO
                    ENDIF
C
C
C     IF A MATCH WAS FOUND, CLEAR OUT ANY REMAINING ASSIGNMENT
C     MESSAGES THAT MAY BE FOUND IN THE ASSIGNMENT FILE.
C
                    IF (II .EQ. MATCH_CODE) THEN
                          CALL CLEAR_FILE (ASSIGN_FILE, PEN, JUNK_BOOLEAN,
     1                                                   JUNK_ATRIB)
                    ENDIF
                ENDIF
C
C     THE RESULT OF THE TWO PREVIOUS CASES IS THE SAME. IF A MATCH HAS
C     BEEN FOUND, THE ASSIGNMENT REPORT IS TO BE FOUND IN THE ARRAY
C     ATRIB, AND THE RADAR INFORMATION IN THAT MESSAGE IS THE RADAR
C     DATA FROM THE DETECTION REPORT.
C
          ELSE
C
C     THIS MESSAGE IS BOTH A DETECTION AND AN ASSIGNMENT. IT IS
```

```
C   SUFFICIENT TO ALLOW ACTION: STILL THE AWAIT FILES MUST BE
C   CLEARED OF ANY OTHER MESSAGES REGARDING THIS PENETRATOR
C
        IF (EXIT_TIME (PEN, CLASS, ITEM) .EQ. 0.0) THEN
          II = DISCARD_CODE
          CALL SEND_HANDOVER_MESSAGE (CLASS, ITEM, ATRIB)
        ELSE
          II = MATCH_CODE
          CALL SEARCH_AWAIT_FILE (ASSIGN_FILE, II,
   1                                      ATRIB, TEMP_ATRIB)
          CALL SEARCH_AWAIT_FILE (DETECT_FILE, II,
   1                                      ATRIB, TEMP_ATRIB)
        ENDIF
     ENDIF
   ENDIF
C
C   AN ENGAGEMENT  MESSAGE WILL BE PROCESSED FURTHER FROM THIS NODE
C   IFF II = MATCH_CODE; IN THIS CASE (AND WHEN THE MESSAGE IS OF
C   ONE OF THE OTHER TYPES) IT MUST BE CALCULATED IN WHAT QUEUE
C   THIS MESSAGE IS TO AWAIT PROCESSING. NOTE THAT CO NODES USE
C   SLAM AWAIT NODES INSTEAD OF QUEUES TO WAIT FOR SERVICE.
C   NONETHELESS, DETERMINE_QUEUE IS STILL USABLE TO FIND THE SLAM
C   FILE NUMBER, SINCE CO FILES HAVE THE SAME LAYOUT AS OTHER
C   FILES, TO A SUFFICIENT LEVEL OF DETAIL
C
C   NOW MAKE SURE THAT ALL NON-ENGAGEMENT MESSAGES AND THE ENGAGEMENT
C   MESSAGES WHICH HAVE FOUND MATCHES (AND HENCE NEED NOT WAIT IN
C   ONE OF THE AWAIT QUEUES) CAN BRANCH TO THE SELECT NODE FOR
C   ASSIGNMENT TO ONE OF THE MAIN CO QUEUES.
C
C
C   *** CODE CHANGE ADDED TO DISCARD ENGAGEMENT MESSAGES ***
C   *** THAT HAVE II = MATCH_CODE (BEGIN PROSECUTION).  IN
C   *** THOSE CASES WHERE THE ID CODE PREVENTS EXECUTION.
C   *** (ID _INDEX = 5,6,7,8), THE ROUTINE MUST SET
C   *** II = DISCARD_CODE SO THE MESSAGE WILL BE DISCARDED
C
C
    IF (PEN_IDENT(PEN) .LT. 5) THEN
C
C   *** CONTINUE ORIGINAL COMATCH CODE
C   *** IF ID IS HOSTILE (3,4)
C
     IF (PEN_IDENT(PEN) .GT. 2) THEN
      IF (MESSAGE_TYPE .NE. CO_ENG_MODEL) THEN
        II = MATCH_CODE
      ENDIF
C
      IF (II .EQ. MATCH_CODE) THEN
        CALL DETERMINE_QUEUE
C
        IF (MESSAGE_TYPE .EQ. CO_ENG_MODEL) THEN
```

```
                ATRIB (CO_DETECTION_RPT) = TRUE
                  ATRIB (ALERT_STATUS)     = TRUE
             ENDIF
          ENDIF
C
C     *** IF ID_INDEX IS UNKNOWN (1 OR 2) THEN ALLOW
C     *** PROSECUTION ONLY IF A FIGHTER CO (ITEM LESS THAN 4)
C     *** ITEM IS THE INDEX FOR THE ORDER OF CO'S
C
        ELSEIF (PEN_IDENT(PEN) .LT. 3) THEN
C
C
C     IF ID CODE WAS ONE OR TWO AND THE CO IS A
C     FIGHTER THEN CONTINUE WITH ORIGINAL CODE
C
          IF (ITEM .LT. 4) THEN
             IF (MESSAGE_TYPE .NE. CO_ENG_MODEL) THEN
                II = MATCH_CODE
             ENDIF
C
             IF (II .EQ. MATCH_CODE) THEN
                CALL DETERMINE_QUEUE
C
                IF (MESSAGE_TYPE .EQ. CO_ENG_MODEL) THEN
                   ATRIB (CO_DETECTION_RPT) = TRUE
                     ATRIB (ALERT_STATUS)     = TRUE
                ENDIF
             ENDIF
          ENDIF
C
        ENDIF
C
      ELSE
C
        II = DISCARD_CODE
C
      ENDIF
C
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS FILE CONTAINS USERF AND SERVICE_TIME, WHICH SERVE TO RETURN
C     THE SERVICE TIME FOR ACTIONS AT ANY STANDARD SERVICE NODE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     USERF IS A STANDARD SLAM FUNCTION, USED FOR CALCULATING DURATIONS
C     OF ACTIVITIES AND ASSIGNING ATTRIBUTES IN ASSIGN NODES. AT THIS
C     POINT, ONLY THE FORMER USAGE OCCURS IN TADZ. USERF (1) IS CALLED
C     TO DETERMINE THE SERVICE TIME IN A SYSTEM NODE. THIS IS ACTUALLY
C     DONE IN FUNCTION SERVICE_TIME, CALLED FROM HERE. AS IN SUBROUTINE
C     EVENT, ICODE IS A SELECTOR VARIABLE, AND ALL VARIABLES AND VALUES
C     USED IN THE CALCULATIONS ARE PASSED IN COMMON BLOCKS.
C
C     ALPHATECH, INC.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      REAL FUNCTION USERF (ICODE)
C
      IMPLICIT NONE
C
C*****INPUT VARIABLES
C
      INTEGER ICODE
C               INPUT CASE SELECTOR
C
C******COMMON BLOCKS AND PARAMETERS
C
      INCLUDE 'DCO:PARAMETER.DIM'
C
C******LOCAL VARIABLES
C
      REAL RETURN_TIME
C               RETURNS VALUE OF SERVICE TIME AT A GIVEN SYSTEM NODE
C
C******EXECUTABLE CODE
C
      IF (ICODE.EQ.SERV_TIME_CODE) THEN
          CALL SERVICE_TIME (RETURN_TIME)
          USERF = RETURN_TIME
      ELSE
          WRITE (ERROR_FILE, *) 'ERROR IN USERF'
          WRITE (ERROR_FILE, *) 'UNKNOWN FUNCTION CODE = ', ICODE
          CALL UERR (FATAL)
      ENDIF
C
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SERVICE_TIME (INPUT PARAMETERS PASSED IMPLICITLY IN THE
C     COMMON BLOCK SCOM1.BLK, BECAUSE OF THE CONSTRAINTS
C     IMPOSED BY SLAM) RETURNS TO THE GENERIC SYSTEM NODE THE SERVICE
C     TIME AS A FUNCTION OF THE NUMBER OF REPORTS FUSED TO PRODUCE
C     THE OUTPUT REPORT. IT INVOLVES MULTIPLE CALLS TO THE SLAM
C     ROUTINES NFIND (FOR SEARCHING FILES TO FIND ENTITIES WITH
C     MATCHING ATTRIBUTES) AND COPY (TO OBTAIN THE ATTRIBUTES OF THESE
C     ENTITIES). REPORTS ON THE ENTRIES DETERMINED BY SERVICE_TIME TO
C     BE EXTRA INFORMATION ARE LOADED INTO THE COMMON BLOCK
C     REMOVAL_RECORDS FOR EVENTUAL DISCARDING IN REMOVE_DUPLICATE_MESSAGES
C     IT DOES NOT WORK TO TRY TO REMOVE THOSE RECORDS FROM THE QUEUES
C     IN THE COURSE OF THIS ROUTINE.
C
C     THIS SUBROUTINE SHOULD BE CALLED ONLY FROM THE SLAM FUNCTION
C     USERF.
C
C     ALPHATECH, INC.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE SERVICE_TIME (RETURN_TIME)
      IMPLICIT NONE
C
C*****OUTPUT PARAMETERS
C
      REAL RETURN_TIME
C               SERVICE TIME RETURNED TO USERF
C
C******COMMON BLOCKS AND PARAMETERS
C
      INCLUDE 'DCO:SLAMPARAM.DIM'
      INCLUDE 'DCO:SCOM1.SLM'
      INCLUDE 'DCO:PARAMETER.DIM'
      INCLUDE 'DCO:REMOVAL.INC'
      INCLUDE 'DCO:S0INIT.PRM'
      INCLUDE 'DCO:S1INIT.PRM'
      INCLUDE 'DCO:S2INIT.PRM'
      INCLUDE 'DCO:C2INIT.PRM'
      INCLUDE 'DCO:C1INIT.PRM'
      INCLUDE 'DCO:C0INIT.PRM'
      INCLUDE 'DCO:SYSCOM.INC'
C
C
C******EXTERNAL FUNCTIONS
C
      INTEGER NFIND
C               USED TO RETURN RANK IN A FILE OF AN ENTRY WITH GIVEN
C               ATTRIBUTES
      INTEGER NNQ
```

```
C               SLAM FUNCTION RETURNS NUMBER OF ENTRIES IN A QUEUE
C
C****** ADDED FOR EXPONENTIAL SERVICETIMES
      REAL EXPON
C               SLAM FUNCTION RETURNS A SAMPLE FROM AN EXPONENTIAL
C               DISTRIBUTION WITH SPECIFIED MEAN
C***************************************
C
      REAL UNFRM
C               SLAM FUNCTION RETURNS A SAMPLE FROM A UNIFORM
C               DISTRIBUTION WITH SPECIFIED ENDPOINTS, USING GIVEN
C               RANDOM NUMBER STREAM.
C     LOGICAL TRACE_ON
C               TRUE IFF THE SLAM TRACE FACILITY IS CURRENTLY ENABLED
      CHARACTER*2 CLASS_STRING
C               CONVERTS INTEGER CLASS REPRESENTATION INTO A STRING
      REAL CURR_TIME
C               OBTAINS THE CURRENT TIME
C
C******LOCAL VARIABLES
C
      REAL TEMP_ATRIB (NUM_ATTRIBUTES)
C               ARRAY FOR TEMPORARY STORAGE OF ATTRIBUTES IN ENTITIES
C               BEING REMOVED FROM THE QUEUES
      INTEGER MESSAGE_COUNT
C               NUMBER OF MESSAGES BEING PROCESSED BY THIS SYSTEM
C               NODE IN THE COURSE OF THIS CALL
      INTEGER FILES (MAX_FILES_TO_CHECK), CURRFILE
C               TEMPORARIES TO STORE THE INDICES OF SLAM QUEUES FOR
C               USE IN SEARCHING FOR DUPLICATE MESSAGES
      LOGICAL MESSAGE_FOUND_IN_QUEUE (MAX_FILES_TO_CHECK)
C               FLAGS TO INDICATE WHETHER MATCHED MESSAGE WAS FOUND IN
C               THE FILES NAMED IN ARRAY FILES.
      INTEGER RANK
C               TEMPORARY VARIABLES USED IN REMOVING MESSAGES FROM QUEUES
C               RANK POINTS TO A MESSAGE WITH MATCHING ATTRIBUTES
C               (== CORRESPONDING TO THE SAME PENETRATOR)
      REAL STORE_SYS_TARGET
C               TEMP USED TO AVOID OVERWRITING CURR_SYS_TARGET
      REAL STORE_ALERT_STATUS
C               SIMILARLY STORES FIRST/CONTINUED REPORT STATUS OF
C               ORIGINAL MESSAGE, AS MAINTAINED IN ATTRIBUTE ALERT_STATUS
      REAL STORE_LAST_CLASS, STORE_LAST_ITEM
C               STORAGE LOCATIONS FOR NAME OF LAST SYSTEM NODE THAT THIS
C               MESSAGE PASSES THROUGH. THIS MUST BE PROPERLY SAVED SO
C               THAT THE ALLOCATION TABLE UPDATING ROUTINE CAN PROPERLY
C               DETERMINE IF THE CURRENT MESSAGE IS THE 'CURRENT SYSTEM
C               MESSAGE' ABOUT THE GIVEN PENETRATOR, AND NOT AN ATTEMPT
C               TO PRODUCE ANOTHER FIRST REPORT (AND HENCE A MULTIPLE
C               ALLOCATION) SOMEWHERE.
      REAL STORE_CO_DETECTION_RPT
C               SAVES AND ACCUMULATES THE VALUE OF CO_DETECTION_RPT. IF
```

```
C                 ANY OF THE MERGED MESSAGES ARE MARKED AS HAVING TRACK
C                 INFORMATION SUFFICIENT TO CONDUCT AN ENGAGEMENT, THEN
C                 THE AGGREGATED MESSAGE SHOULD
      INTEGER C_SIDE_CLASS, C_SIDE_ITEM
C                 ALSO POTENTIALLY USED IN GENERATING THE APPROPRIATE
C                 ENTRIES FOR LAST_CLASS, LAST_ITEM IN THE OUTGOING MESSAGE
      INTEGER INT_LAST_CLASS
C                 NEAREST INTEGER TO ATRIB (LAST_CLASS); USED FOR COMPARISONS
      INTEGER I, J
C                 COUNTERS
      INTEGER PEN
C                 PENETRATOR TAIL NUMBER
      INTEGER CLASS, ITEM
C                 SYSTEM NODE AT WHICH SERVICE IS BEING PERFORMED
      REAL TEMP_CURR_TIME
C                 SAVES THE RETURNED VALUE OF THE CURRENT TIME
C
C     THE FOLLOWING VARIABLES ARE USED TO STORE THE PARAMETERS EXTRACTED
C     FROM THE VARIOUS NODE INITIALIZATION BLOCKS. THESE VALUES ARE USED
C     IN THE FINAL CALCULATION OF THE SERVICE TIME.
C
      REAL FIRST_TIME
C                 TIME FOR A FIRST REPORT TO BE PROCESSED AT THE GIVEN NODE
      REAL CONTINUED_TIME
C                 DITTO FOR A CONTINUED REPORT
      REAL FUSION_TIME
C                 ADDITIONAL TIME REQUIRED FOR EACH ADDITIONAL MESSAGE TO
C                 BE FUSED WITH THE PREVIOUSLY FOUND MESSAGES REGARDING
C                 THE SAME PENETRATOR
      REAL FIRST_MESSAGE_TIME
C                 TIME REQUIRED TO SERVICE THE FIRST MESSAGE, DEPENDING ON
C                 WHETHER IT IS A FIRST OR A CONTINUED REPORT, INCLUDING
C                 STOCHASTIC EFFECTS
      REAL FUSION_SUM
C                 SUM TOTAL OF TIME REQUIRED TO PERFORM FUSION ON
C                 ADDITIONAL MESSAGES BEING FUSED WITH ORIGINAL MESSAGE--
C                 ALSO STOCHASTIC
      REAL MULTIPLIER
C                 IF THE NODE PERFORMING THE SERVICE IS NOT PERFORMING ITS
C                 ORIGINAL FUNCTION, IT CAN BE EXPECTED TO TAKE LONGER TO
C                 COMPLETE SERVICE. MULTIPLIER IS USED TO ADD THAT ADDITIONAL
C                 TERM TO THE EXPRESSION FOR SERVICE TIME.
C
C******EXECUTABLE CODE
C
C     ONE SIDE EFFECT OF THIS FUNCTION IS TO POSSIBLY RESET THE XX
C     VARIABLE ASSOCIATED WITH THIS QUEUE TO TALLY THE BUSY STATUS
C     OF THE QUEUE THIS MESSAGE WAS JUST TAKEN FROM, TO CALCULATE
C     THE PROBABILITY OF QUEUEING FOR A MESSAGE OF THIS TYPE
C
      IF (NNQ (NINT (ATRIB (MOST_RECENT_QUEUE))) .EQ. 0) THEN
          XX (NINT (ATRIB (MOST_RECENT_QUEUE))) = 0
      ENDIF
C
C     THE MESSAGE COMING OUT OF THIS PROCESS IS THE MESSAGE OF
C     HIGHEST PRIORITY FOUND IN THE CURRENT C3 NODE.
C     THE NUMBER OF MESSAGES FUSED TO GENERATE THIS MESSAGE IS
C     A FACTOR IN DETERMINING THE AGGREGATE SERVICE TIME.
C     THE MESSAGE, HOWEVER, MUST INDICATE THAT THERE IS A CURRENT
C     SYSTEM TARGET IF ANY OF THE FUSED MESSAGES SAY THERE IS, EVEN
C     IF THE LATEST MESSAGE DOES NOT SAY SO. SINCE ALL MESSAGES
C     REGARDING SYSTEM TARGETS ARE OF THE HIGHEST PRIORITY, IF THE
C     INPUT MESSAGE TO THIS NODE (WITH ATTRIBUTES IN ATRIB) DOES NOT
C     STATE THAT THE SYSTEM IS ENDANGERED, THEN NONE OF THE MESSAGES
C     REGARDING THIS PENETRATOR WILL. THIS ATTRIBUTE IS STORED IN
C     STORE_SYS_TARGET, TO BE RESTORED TO THE FINAL MESSAGE UPON
C     LEAVING THIS PROCESS. SIMILARLY, IF THE REPORT BEING SERVICED
C     IS NEITHER AN INDICATION OF IMPENDING DANGER TO THE SYSTEM NOR
C     A FIRST REPORT, THEN ALL THE OTHER REPORTS MERGED WITH IT HERE
C     WILL ALSO BE CONTINUED REPORTS.
C
C     IT IS ALSO NECESSARY TO SAVE THE LAST CLASS AND ITEM THAT THIS
C     MESSAGE PASSED THROUGH, FOR THOSE VALUES ARE NEEDED FOR THE
C     PROPER MAINTAINENCE OF THE ALLOCATION TABLE, AS WILL BE SEEN
C     BELOW.
C
      STORE_SYS_TARGET   = ATRIB (CURR_SYS_TARGET)
      STORE_ALERT_STATUS = ATRIB (ALERT_STATUS)
      STORE_LAST_CLASS   = ATRIB (LAST_CLASS)
      STORE_LAST_ITEM    = ATRIB (LAST_ITEM)
      STORE_CO_DETECTION_RPT = ATRIB (CO_DETECTION_RPT)
      C_SIDE_CLASS = 0
      C_SIDE_ITEM  = 0
C
      PEN           = NINT (ATRIB (TRUE_PEN_NO))
      CLASS         = NINT (ATRIB (CURRENT_CLASS))
      ITEM          = NINT (ATRIB (CURRENT_ITEM))
      MESSAGE_COUNT = NINT (ATRIB (CURR_NUM_MESSAGES))
C
C     DETERMINE WHAT FILES MESSAGES ABOUT THIS PENETRATOR MAY BE IN
C
      CALL GET_FILES (CLASS, ITEM, FILES, ATRIB)
C
C
      DO I = 1, MAX_FILES_TO_CHECK
         MESSAGE_FOUND_IN_QUEUE (I) = .FALSE.
      ENDDO
C
      DO I = 1, MAX_FILES_TO_CHECK
         CURRFILE = FILES (I)
C
C     CHECK EACH OF THESE FILES FOR FURTHER MESSAGES REGARDING THIS
C     PENETRATOR.
C
```

```
          IF (CURRFILE .NE. 0) THEN
               RANK = 1
C
C     START SEARCHING FROM THE FIRST ENTRY
C
               DOWHILE ((RANK .NE. 0) .AND. (RANK .LE. NNQ (CURRFILE)))
                   RANK = NFIND (RANK, CURRFILE, TRUE_PEN_NO, 0,
     1                                          ATRIB (TRUE_PEN_NO), 0)
C
C     THESE PARAMETERS CAUSE SLAM TO LOOK FOR AN ENTRY IN FILE
C     CURRFILE WHOSE ATTRIBUTE TRUE_PEN_NO IS AN EXACT MATCH FOR THE
C     SAME ATTRIBUTE IN THE ARRAY ATRIB, I.E. THE MESSAGE REFERS TO
C     THE SAME PENETRATOR. THE REMAINDER OF THIS DO LOOP IS EXECUTED
C     ONLY IF SUCH A MATCH WAS FOUND.
C
                   IF (RANK .NE. 0) THEN
                         MESSAGE_FOUND_IN_QUEUE (I) = .TRUE.
                         CALL COPY (RANK, CURRFILE, TEMP_ATRIB)
                         MESSAGE_COUNT = MESSAGE_COUNT + NINT (TEMP_ATRIB
     1                                        (CURR_NUM_MESSAGES))
C
C     SAVE THE LAST SYSTEM NODE IF THAT NODE WAS A C SIDE NODE.
C
                         INT_LAST_CLASS = NINT (TEMP_ATRIB (LAST_CLASS))
                         IF ((INT_LAST_CLASS .EQ. C2) .OR.
     1                                 (INT_LAST_CLASS .EQ. C1) .OR.
     2                                 (INT_LAST_CLASS .EQ. C0)) THEN
                             C_SIDE_CLASS = INT_LAST_CLASS
                             C_SIDE_ITEM  = NINT (TEMP_ATRIB (LAST_ITEM))
                         ENDIF
C
                         IF (TEMP_ATRIB (CO_DETECTION_RPT) .EQ. TRUE)
     +                                                           THEN
                              STORE_CO_DETECTION_RPT = TRUE
                         ENDIF
C
                         IF (TEMP_ATRIB (LAST_REPORT_TIME) .GT.
     1                                       ATRIB (LAST_REPORT_TIME)) THEN
C
C     THE REPORT JUST RETRIEVED FROM THE QUEUE IS MORE RECENT THAN
C     THE PREVIOUS ONE. REPLACE THE OLDER REPORT WITH THE NEWER ONE

                             DO J = 1, NUM_ATRIB_USED
                                   ATRIB (J) = TEMP_ATRIB (J)
                             ENDDO
                         ENDIF
C
C     GO ON TO THE NEXT MESSAGE IN THE FILE FOR THE NEXT TIME THROUGH
C     THE LOOP
C
                         RANK = RANK + 1
C
```

```
                    ENDIF
                ENDDO
C
C     NOW CURRFILE MAY HAVE BEEN EMPTIED OF MESSAGES IN THE PROCESS
C     OF PERFORMING THIS MERGER. IF SO, THEN RESET THE XX VARIABLE
C     WHICH INDICATES WHETHER THERE IS A MESSAGE IN CURRFILE. (IT
C     CAUSES NO HARM TO PERFORM THIS RESET IF THE FILE WAS INDEED
C     EMPTY ALREADY.)
C
                IF (NNQ (CURRFILE) .EQ. 0) THEN
                    XX (CURRFILE) = 0.0
                ENDIF
          ENDIF
      ENDDO
C
C     AT THIS POINT MESSAGE_COUNT CONTAINS A COUNT OF THE NUMBER OF
C     MESSAGES FOUND THAT REFERRED TO THE TARGET CURRENTLY UNDER
C     CONSIDERATION. THE FILE(S) IN WHICH ANY DUPLICATE MESSAGE IS
C     LOCATED ARE INDICATED BY .TRUE. ENTRIES IN MESSAGE_FOUND_IN_QUEUE.
C     THE ATTRIBUTES OF THE MOST RECENT MESSAGE (WHICH IS NOT
C     NECESSARILY THE MESSAGE THAT WAS ORIGINALLY BEING SERVED) ARE
C     LOCATED IN ARRAY ATRIB, AND THE VARIABLE STORE_SYS_TARGET
C     CONTAINS A VALUE WHICH INDICATES WHETHER ANY OF THE MESSAGES
C     INDICATED THAT A SYSTEM COMPONENT WAS IMMINENTLY UNDER ATTACK.
C     NOW CLEAN UP THESE VALUES.
C
      DO I = 1, MAX_FILES_TO_CHECK
          IF (MESSAGE_FOUND_IN_QUEUE (I)) THEN
C
C     INDICATE IN COMMON BLOCK PASSED TO REMOVAL SYSTEM NODE THAT THIS
C     FILE MUST BE SEARCHED FOR ELEMENTS MATCHING THE PENETRATOR
C     NUMBER. IT WOULD INDEED BE MORE NATURAL TO SIMPLY DISCARD ALL
C     THE DUPLICATED MESSAGES HERE, BUT DOING SO IN THE CONTEXT OF THE
C     SLAM FUNCTION USERF (WHERE SERVTIME IS CALLED) CAUSES THE
C     POINTER MFA (WHICH LOCATES THE HEAD OF THE FREE MESSAGE SLOT
C     LIST IN THE COMMON ARRAY NSET/QSET) TO BE DISLOCATED, AND SLAM
C     EITHER DIES FAIRLY SHORTLY THEREAFTER OR ELSE YIELDS TRUE
C     NONSENSE FOR THE NEXT ACTIONS.
C
                NUM_FILES_TO_SEARCH = NUM_FILES_TO_SEARCH + 1
                IF (NUM_FILES_TO_SEARCH .GT. MAX_FILES_TO_MATCH) THEN
                    WRITE (ERROR_FILE, *) 'ERROR IN SERVICE_TIME'
                    WRITE (ERROR_FILE, *) 'OVERFLOW OF LIST OF FILES ',
     +                                    'MARKED FOR DELETION'
                    WRITE (ERROR_FILE, *) 'FILES NAMED TO BE SEARCHED'
                    WRITE (ERROR_FILE, *) (FILES_TO_SEARCH (J), J = 1,
     +                                    MAX_FILES_TO_MATCH)
                    CALL UERR (FATAL)
                ELSE
                    FILES_TO_SEARCH (NUM_FILES_TO_SEARCH) = FILES (I)
                    ATRIB_VALUE_TO_MATCH (NUM_FILES_TO_SEARCH) =
     1                          ATRIB (TRUE_PEN_NO)
```

```
                ENDIF
          ENDIF
      ENDDO
C
      ATRIB (CURR_SYS_TARGET) = STORE_SYS_TARGET
      ATRIB (ALERT_STATUS) = STORE_ALERT_STATUS
      IF (ATRIB (HANDOVER_MESSAGE) .EQ. TRUE) THEN
          ATRIB (CO_DETECTION_RPT) = FALSE
C
C     ... SINCE A HANDOVER MESSAGE IS BEING PASSED UP THE CHAIN OF
C     COMMAND, THE TRACK INFORMATION THAT MAY BE ASSOCIATED WITH THIS
C     MESSAGE CAN NO LONGER BE RELEVANT; IT WASN'T POSSIBLE TO PERFORM
C     PROSECUTION IN THE IMMEDIATE NEIGHBORHOOD
C
      ELSE
          ATRIB (CO_DETECTION_RPT) = STORE_CO_DETECTION_RPT
      ENDIF
C
C     NOW RESET THE LAST_CLASS AND LAST_ITEM ATTRIBUTES, SO THAT ANY
C     MESSAGE THAT CAME FROM THE C SIDE HAS ITS SOURCE NODE PRESERVED.
C     THE ORDERING IS CHOSEN SO THAT IF THE ORIGINAL MESSAGE BEING
C     PROCESSED (I.E. THE MESSAGE AT THIS NODE OF THE HIGHEST
C     PRIORITY) CAME FROM THE C SIDE, THEN ITS SOURCE NODE IS THE
C     SOURCE NODE OF THE ENSUING MESSAGE. OTHERWISE IF ANY OTHER
C     MESSAGE CAME FROM THE C SIDE THEN USE ITS SOURCE NODE, AS SAVED
C     IN C_SIDE_CLASS AND C_SIDE_ITEM. IN ANY OTHER CASE IT SUFFICES
C     TO USE THE MOST RECENT MESSAGE'S LAST CLASS/ITEM.
C
      INT_LAST_CLASS = NINT (STORE_LAST_CLASS)
      IF ((INT_LAST_CLASS .EQ. C2) .OR. (INT_LAST_CLASS .EQ. C1)
     1                    .OR. (INT_LAST_CLASS .EQ. C0)) THEN
          ATRIB (LAST_CLASS) = STORE_LAST_CLASS
          ATRIB (LAST_ITEM)  = STORE_LAST_ITEM
      ELSEIF (C_SIDE_CLASS .NE. 0) THEN
          ATRIB (LAST_CLASS) = C_SIDE_CLASS
          ATRIB (LAST_ITEM)  = C_SIDE_ITEM
      ENDIF
C
C     COLLECT THE APPROPRIATE PARAMETERS FOR SERVICE BASE TIMES FROM
C     THE VARIOUS INITIALIZATION COMMON BLOCKS, DEPENDING ON THE
C     CLASS OF NODE PERFORMING THE SERVICE. ONLY FOR S2 AND C2 NODES
C     CAN MULTIPLIER TERMS OTHER THAN 1 APPLY.
C
      IF (CLASS .EQ. S0) THEN
          FIRST_TIME     = FIRST_S0_SERVICE  (ITEM)
          CONTINUED_TIME = CONT_S0_SERVICE   (ITEM)
          FUSION_TIME    = FUSION_S0_SERVICE (ITEM)
          MULTIPLIER     = 1
      ELSEIF (CLASS .EQ. S1) THEN
          FIRST_TIME     = FIRST_S1_SERVICE  (ITEM)
          CONTINUED_TIME = CONT_S1_SERVICE   (ITEM)
          FUSION_TIME    = FUSION_S1_SERVICE (ITEM)
```

```
         MULTIPLIER     = 1
      ELSEIF (CLASS .EQ. S2) THEN
         FIRST_TIME     = FIRST_S2_SERVICE  (ITEM)
         CONTINUED_TIME = CONT_S2_SERVICE   (ITEM)
         FUSION_TIME    = FUSION_S2_SERVICE (ITEM)
         IF (ACTUAL_S2_CLASS .EQ. S1) THEN
               MULTIPLIER = 1 / S1_S2_MULTIPLIER (ACTUAL_S2_ITEM)
         ELSE
               MULTIPLIER = 1
         ENDIF
      ELSEIF (CLASS .EQ. C2) THEN
         FIRST_TIME     = FIRST_C2_SERVICE  (ITEM)
         CONTINUED_TIME = CONT_C2_SERVICE   (ITEM)
         FUSION_TIME    = FUSION_C2_SERVICE (ITEM)
         IF (ACTUAL_C2_CLASS .EQ. C1) THEN
               MULTIPLIER = 1 / C1_C2_MULTIPLIER (ACTUAL_C2_ITEM)
         ELSE
               MULTIPLIER = 1
         ENDIF
      ELSEIF (CLASS .EQ. C1) THEN
         FIRST_TIME     = FIRST_C1_SERVICE  (ITEM)
         CONTINUED_TIME = CONT_C1_SERVICE   (ITEM)
         FUSION_TIME    = FUSION_C1_SERVICE (ITEM)
         MULTIPLIER     = 1
      ELSEIF (CLASS .EQ. C0) THEN
         FIRST_TIME     = FIRST_C0_SERVICE  (ITEM)
         CONTINUED_TIME = CONT_C0_SERVICE   (ITEM)
         FUSION_TIME    = FUSION_C0_SERVICE (ITEM)
         MULTIPLIER     = 1
      ENDIF
C
C     NOW SET THE RETURN TIME BASED ON THE VALUES HERE
C
      IF ((ATRIB (CURR_SYS_TARGET) .EQ. TRUE) .OR.
     1                              (ATRIB (ALERT_STATUS) .EQ. TRUE)) THEN
         FIRST_MESSAGE_TIME = EXPON(FIRST_TIME,1)
C
C******** NOTE THE CHANGE TO EXPONENTIAL FIRST SERVICE TIMES
C******** THE ORIGINAL COD IS COMMENTED OUT BELOW
C
C        FIRST_MESSAGE_TIME = UNFRM
C     +                    (LOW_LIM_MULT * FIRST_TIME,
C     +                     HIGH_LIM_MULT * FIRST_TIME, 1)
C
      ELSE
         FIRST_MESSAGE_TIME = UNFRM
     +                    (LOW_LIM_MULT * CONTINUED_TIME,
     +                     HIGH_LIM_MULT * CONTINUED_TIME, 1)
      ENDIF
C
C     IF (FIRST_MESSAGE_TIME .GT. 150) THEN
C       FIRST_MESSAGE_TIME = 150
```

```
C     ENDIF
C
      FUSION_SUM = 0.0
      DO I = 2, MESSAGE_COUNT
                FUSION_SUM = FUSION_SUM + UNFRM
     +                              (LOW_LIM_MULT * FUSION_TIME,
     +                              HIGH_LIM_MULT * FUSION_TIME, 1)
      ENDDO
C
      RETURN_TIME = MULTIPLIER * (FIRST_MESSAGE_TIME + FUSION_SUM)
D     IF (TRACE_ON ()) THEN
D         WRITE (TRACEFILE, *)
D         WRITE (TRACEFILE, *) '------------------------------'
D         WRITE (TRACEFILE, *)
D         TEMP_CURR_TIME = CURR_TIME (1)
D         WRITE (TRACEFILE, *) 'CURRENT TIME = ', TEMP_CURR_TIME
D         WRITE (TRACEFILE, *) 'BEGINNING SERVICE ON MESSAGE AT ',
D    +                              'NODE ', CLASS_STRING (CLASS), ' ', ITEM
D         WRITE (TRACEFILE, *) 'PENETRATOR TAIL NUMBER = ', PEN
D         WRITE (TRACEFILE, *)
D         IF ((ATRIB (CURR_SYS_TARGET) .EQ. TRUE) .OR.
D    1                              (ATRIB (ALERT_STATUS) .EQ. TRUE)) THEN
D               WRITE (TRACEFILE, *) 'FIRST REPORT BEING PROCESSED'
D         ELSE
D               WRITE (TRACEFILE, *) 'CONTINUED REPORT BEING PROCESSED'
D         ENDIF
D         IF (MESSAGE_COUNT .GT. 1) THEN
D               WRITE (TRACEFILE, *) MESSAGE_COUNT, ' MESSAGES MERGED ',
D    +                                        'TO PRODUCE THIS REPORT'
D         ENDIF
D         WRITE (TRACEFILE, *)
D         WRITE (TRACEFILE, *) 'FINAL SERVICE TIME = ', RETURN_TIME
D         WRITE (TRACEFILE, *) 'THIS SERVICE WILL BE COMPLETED AT ',
D    +                                        'TIME ', TEMP_CURR_TIME + RETURN_TIME
D         WRITE (TRACEFILE, *)
D         WRITE (TRACEFILE, *) '------------------------------'
D         WRITE (TRACEFILE, *)
D     ENDIF
C
      RETURN
      END
```

APPENDIX B: TADZ/CRC USER'S MANUAL

TADZ is a complicated model consisting of both SLAM and FORTRAN coding. For the purposes of the CRC representation used in this tactical scenario, the user need only be concerned with the "input files" and implementation procedures for TADZ. This appendix will describe the input files required to run the model along with a step by step implementation guide. The basic format for these worksheets was created by Mr. Rick Bowman, FTD/TQC [BOWMAN, 1985].

Input Files

For TADZ to model even the simplest of tactical scenarios, a minimum of thirty-one input files must be created by the user. There are four major classes of input data:

1. geography of the air defense zone (ADZ)
2. system order of battle
3. penetrator order of battle
4. control of the simulation and outputs

This user's guide will essentially abbreviate the information that is found in the TADZ User's Guide [Merriman et al, 1984]. For more detailed explaination or descriptions of these input files, refer to that reference. Several of the input files, although needed to run the model, do not require user manipulation for the tactical scenario used here. These files will *not* be described in this appendix, however the examples of *each* type of input file are provided in Appendix C.

SYSLIM.DAT. This file is the first file read, since it contains dimensioning data for several other input files. SYSLIM sets the number of elements in the C^3 system being modeled, to include the number of "nodes" in the network, the number of penetrator types, and the number of paths or routes used by those penetrators. All these parameters are bounded by limits which are set in the global parameter file. Care should be taken not to exceed these limits, although the current limits are set well above the numbers required to model the scenario for the thesis. Another "caveat": it is *critical* to keep the parameters consistent across the input files, as these numbers are used in the loading of the common blocks and arrays used in the simulation.

PARAMETER	VALUE
NUMBER OF R^0's (NUMRADAR)	
NUMBER OF S^0's (NUMS0)	
NUMBER OF S^1's (NUMS1)	

NUMBER OF S^2's (NUMS2) MUST = 1

NUMBER OF C^2's (NUMC2) MUST = 1

NUMBER OF C^1's (NUMC1)

NUMBER OF C^0's (NUMC0)

NUMBER OF PENETRATOR TYPES (NUMPENTYPES)

NUMBER OF PENETRATOR PATHS (NUMPATHS)

ROx.DAT. The R^O files include site and type information for each radar set modeled in the system. In this scenario, there are 7 R^O's modeled: four SAM locations and three EW radars. The EW radars are co-located and are needed to model a controller for each CAP point within the ADZ. The documentation in the following example describes the required information in more detail.

PARAMETER	VALUE
LOCATION: LAT, LONG, ALTITUDE (ROLOCATION) TENTHS OF DEGREES, METERS *ALTITUDE INCLUDES ANTENNA HEIGHT	
IDENTIFICATION OF RADAR TYPE: (ROTYPE) RELATIVE POSITION IN EXEC FILE E.G. THE FIFTH RADAR ON THE LIST IS RADAR TYPE 5	
NUMBER OF MASKING ANGLES WHERE TERRAIN MASKING OCCURS: (NUMMASKANGLES)	
MASK ANGLE SPECIFICATIONS: (MASKANGLES) AZIMUTH AND ELEVATION FOR THE ANGLES ABOVE	AZ ELEV
STRUCTURE/HARDNESS DESIGNATION FOR RADAR: (ROSTRUCTURE) NOT USED, BUT A NUMBER MUST BE INPUT!	5

NUMBER OF S^O SUPERVISORS FOR THIS R^O:
(NOROSO) CAN ONLY HAVE ONE!

INDEX OF THE S^O SUPERVISOR:
(IROSO) RELATIVE POSITION IN EXEC FILE

TIME TO PROCESS FIRST REPORTS (SECONDS):
(FRSTROSERVICE)

TIME TO PROCESS CONTINUED REPORTS (SECONDS):
(CONTROSERVCE)

TIME TO FUSE TWO MESSAGES:
(FUSNROSERVICE)

<u>SOx.DAT</u>. These files contain the descriptions of the S^O's in the network. The information required is quite similar to the ROx.DAT files. For the tactical scenario chosen for this project, the number of S^O's is equal to the number of R^O's. Again, the explainations in the example file should clarify any questions that arise. One caution in this file: when describing the connectivity from the S^O, this node can be linked to either an S^1 or an S^2 node but <u>not</u> both. The region of responsibility for an S^O is defined implicitly as the union of the regions of all the radars that report to it.

PARAMETER	VALUE
LOCATION: LAT, LONG, ALT TENTHS OF DEGREES, METERS (SOLOCATION)	
STRUCTURE/HARDNESS DESIGNATION: (SOSTRUCTURE)	1
NUMBER OF S^1 SUPERVISORS FOR THIS S^O (NOSOS1): MUST BE EITHER 0 OR 1 CAN BE LINKED WITH AN S^1 OR S^2, BUT NOT BOTH!	
INDEX OF S^1 SUPER (ISOS1): RELATIVE POSITION IN EXEC FILE LEAVE BLANK IF NO S^1 SUPERS	

```
NUMBER OF C^0 NODES CONNECTED TO THIS S^0:
    (NOSOC0) MUST BE EITHER 0 OR 1. CAN BE LINKED
    TO A C^0 OR C^1 BUT NOT BOTH!


INDEX OF C^0 SUPER (ISOC0):
    RELATIVE POSITION IN EXEC FILE
    BLANK IF NO C^0 LINK


NUMBER OF C^1 CONNECTIONS (NOSOC1):
    MUST BE 0 OR 1


INDEX OF C^1 SUPER (ISOC1):


NUMBER OF S^2 SUPERVISORS (NOSOS2):
    MUST BE 0 OR 1


INDEX OF S^2 SUPER (ISOS2):
    LEAVE BLANK IF NONE


TIME TO PROCESS FIRST REPORTS (FRSTSOSERVICE):
    IN SECONDS


TIME TO PROCESS CONTINUED REPORTS
(CONTSOSERVICE):  IN SECONDS


TIME TO FUSE TWO MESSAGES ON SAME TARGET
(FUSNSOSERVICE): IN SECONDS
```

S1xx.DAT. This file contains much the same information required in the S0x.DAT files. The S^1 node is found one level "up" the heirarchy in the C^3 network. The last two varibles in this type of file (S1_S2_SELECT and S1_S2_MULTIPLIER) are used when the network must be reconfigured when nodes are destroyed by the attacking penetrators. This effort did not employ this capability of TADZ. The interested user is referred to the TADZ User's Guide for more information on this subject [Merriman et al, 1984:69,70].

PARAMETER	VALUE
LOCATION (S1LOCATION): LAT, LONG, ELEV.	
STRUCTURE/HARDNESS (S1STRUCTURE):	1
NUMBER OF S^2 SUPERS FOR THIS S^1: (NOS1S2) MUST BE 0 OR 1	
INDEX OF THE S^2 SUPER (IS1S2): RELATIVE POSTITION IN EXEC FILE	
NUMBER OF C^1 LINKS TO THIS S^1: (NOS1C1) MUST BE 0 OR 1	
INDEX OF THE C^1 SUPER (IS1C1): RELATIVE POSITION IN EXEC FILE	

PROCESS TIME FOR FIRST REPORTS:
(FRSTS1SERVICE) IN SECONDS

TIME FOR CONTINUED REPORTS:
(CONTS1SERVICE) IN SECONDS

TIME TO FUSE TWO REPORTS (FUSNS1SERVICE):

RELATIVE PRIORITY FOR THIS S^1 TO REPLACE 0
S^2 (S1S2SELECT):
INPUT REQUIRED, BUT NOT MODELED IN THIS
SCENARIO

S^1 PERFORMANCE CAPABILITY AS S^2: 1
(S1S2MULTIPLIER) % OF ORIGINAL S^2
CAPABLITY

S2xx.DAT. The S^2 nodes are yet another step up the network "chain". These nodes are described in the same manner as the S^0 and S^1 nodes. The region of responsibility of the S^2 is the union of the regions defined in the S^0 and S^1 input files "below" or reporting to it.

PARAMETER	VALUE
LOCATION (S2LOCATION): TENTHS OF DEGREES, METERS	
STRUCTURE/HARDNESS (S2STRUCTURE): NOT USED IN THIS SCENARIO, BUT MUST INPUT A NUMBER!	1
NUMBER OF C^2 SUPERS FOR THIS S^2: (NOS2C2) MUST BE 1!	1
INDEX OF C^2 SUPERVISOR (IS2C2): RELATIVE POSITION IN EXEC FILE	1
FIRST REPORT PROCESS TIME (FRSTS2SERVICE):	
CONTINUED REPORT SERVICE TIME (CONTS2SERVICE):	
FUSION TIME FOR MESSAGES (FUSNS2SERVICE):	

C2Ox.DAT. This file contains the information describing this first or upper level of the "command" structure of the network. An explicit geographic area of responsibility is given in this file. This area is actually the outer boundary or overall ADZ region. The elevations of the boundary points are not required here as the airspace is assumed to extend from the ground to infinity above the defined region. Note that there are no network connections described in this file. All required connections are given in the appropriate S^2, C^1, and C^0 input files.

PARAMETER	VALUE	
LOCATION (C2LOCATION): LAT, LONG, ALTITUDE TENTHS OF DEGREES AND METERS		
STRUCTURE/HARDNESS DESIGNATION (C2STRUCTURE): NOT USED, BUT INPUT IS REQUIRED	1	
NUMBER OF C^2 BOUNDARY POINTS (C2GBOUND): MAX OF 20 POINTS TO DESCRIBE THE AREA OF RESPONSIBILITY OF THE C^2		
LAT, LONG OF BOUNDARY POINTS (C2BOUNDARY): TENTHS OF DEGREES	LAT	LONG
FIRST C^2 SERVICE TIME (FRSTC2SERVICE): IN SECONDS--TIME TO MAKE ASSIGNMENT MODELED AS 0 IN THIS SCENARIO	0	

CONTINUED C^2 SERVICE TIME (CONTC2SERVICE): IN SECONDS	0
TIME TO FUSE TWO MESSAGES ON SAME TRACK: (FUSNC2SERVICE) IN SECONDS	0

<u>C1xx.DAT</u>. This file contains the information describing the command level just above the missile sites. In the TACS scenario this node represents the ADLO located in the CRC.

PARAMETER	VALUE	
LOCATION (C1LOCATION): LAT, LONG, ALT (TENTHS OF DEGREES)		
STRUCTURE (C1STRUCTURE) NOT USED IN THE TACTICAL SCENARIO	1	
NUMBER OF BOUNDARY POINTS (C1GBOUND) (MAX OF 20)		
LAT, LONG OF BOUNDARY POINTS (C1BOUNDARY) (TENTHS OF DEGREES)	LAT:	LONG:
NUMBER OF C^2 SUPERVISORS (NOC1C2) (EITHER 0 OR 1)	1	
INDEX OF C^2 SUPER (IC1C2) (RELATIVE POSITION IN EXEC FILE LEAVE BLANK IF NONE)	1	
TIME TO EVAL ASSIGNMENT STATUS AND MAKE ASSIGNMENT OR HANDOVER (SECONDS) (FRSTC1SERVICE)	0	
TIME TO SEND TRACK DATA TO C^0s UNDER C^1 CONTROL (SECS) (CONTC1SERVICE)	0	

TIME TO FUSE MESSAGES ON SAME TRACK (FUSNC1SERVICE)	0
RELATIVE PRIORITY FOR THIS C^1 TO REPLACE C^2 (0 = NOT ELIGIBLE) (C1C2SELECT)	0
C^1 PERFORMANCE AS C^2 (% OF ORIGINAL C^2 CAPABILITY) (C1C2MULTIPLIER)	1

PATHSI. This file contains the path data for the routing of the enemy penetrators. The different routes are indexed sequentially and are addressed by this index number in other parts of the code e.g. (PENTIM.DAT).

PARAMETER	VALUE
NUMBER OF POINTS IN PATH/ROUTE MAX OF 25 POINTS (NPOINTSI)	
LAT, LONG.ALT OF EACH POINT ALONG PATH (TENTHS OF DEGREES, METERS) (IPOINTSI)	

** FIRST POINT OF THE PATH MUST LIE OUTSIDE OF C^2 BOUNDARY AND ALL RADAR COVERAGE, INCLUDING LINE OF SIGHT!!

* THESE INPUTS ARE REPEATED FOR THE DESIRED NUMBER OF PATHS. ENSURE THE NUMBER OF PATHS EQUALS THAT SET IN SYSLIM.DAT.

PENSI.DAT. This file contains the descriptions of the different penetrator types being modeled in the scenario. Examples: 1=Foxbat, 2=Flogger, 3=Bear. Maximum of 4 types may be decribed. The relative position of the penetrator in this file determines the prosecution priority of that type e.g. type 1 is the highest priority.

PARAMETER	VALUE
INDEX/ID OF PENETRATOR TYPE (PEN_TYPE)	1
PENETRATOR VELOCITY (M/SEC) (PENSPEED)	
PENETRATOR RADAR CROSS SECTION (M^2) (PENCROSS)	
NUMBER OF WARHEADS BY TYPE CARRIED BY THE PENETRATOR. MUST HAVE 5 ENTRIES EVEN IF LESS THAN 5 WARHEADS DEFINED IN WRHEAD.DAT FILE. (PENBOMBS) E.G. 4,8,0,0,0	
ECM XMTR POWER (WATTS/MHZ) TADZ REQUIRES TWO VALUES. 2nd NOT CURRENTLY USED (ECMPOWER)	

** REPEAT THE ABOVE VALUES FOR UP TO 4 PEN_TYPES. NUMBER DEFINED MUST AGREE WITH SYSLIM.DAT.

PKBLK.DAT. This file contains the information for the short range PK's of the penetrators against the fighter interceptors (FI's). The "matrix" size will be the number of penetrator types by the number of FI types as defined in the PENSI.DAT and WEPARA.DAT INPUT files respectively.

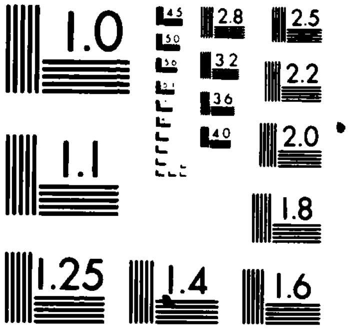
PARAMETER		VALUE		
PROBABILITY OF KILL MATRIX (PEN_ON_UNIT)		FI-1	FI-2	FI-3 ......
	PEN-1			
	PEN-2			
	PEN-3			
	PEN-4			

* VALUES IN THE MATRIX RANGE FROM 0 TO 1.0. THESE VALUES SHOULD ONLY BE NON-ZERO IF THE PENETRATOR ACTUALLY HAS THE CAPABILITY TO FIRE SHORT RANGE GUNS OR MISSILES AGAINST FI's.

WRHEAD.DAT. This data file contains the specifications for the warheads for the penetrators. These warhead types can be defined whether or not the PK's are greater than zero in the PKBLK.DAT file.

PARAMETER	VALUE
NUMBER OF PENETRATOR WARHEAD TYPES MAX OF 5. (NUMWRHEADS)	
WARHEAD RANGE IN METERS (WHRANGE)	
WARHEAD CLASS (1=GRAVITY BOMB; 2=SELF-PROPELLED) (NWHTYPE)	
WARHEAD SPEED IN M/SEC (WHSPEED)	
WARHEAD YIELD IN KILOTONS (WHYIELD)	

AD-A172 447   A SIMULATION ANALYSIS OF AN AUTOMATED IDENTIFICATION   3/3
PROCESSOR FOR THE TA..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..
UNCLASSIFIED   R C MACFARLANE ET AL. JUN 86   F/G 17/7   NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

<u>WRHEAD.DAT</u>. This data file contains the specifications for the warheads for the <u>penetrators</u>. These warhead types can be defined whether or not the PK's are greater than zero in the PKBLK.DAT file.

PARAMETER	VALUE
NUMBER OF PENETRATOR WARHEAD TYPES MAX OF 5. (NUMWRHEADS)	
WARHEAD RANGE IN METERS (WHRANGE)	
WARHEAD CLASS (1=GRAVITY BOMB; 2=SELF-PROPELLED) (NWHTYPE)	
WARHEAD SPEED IN M/SEC (WHSPEED)	
WARHEAD YIELD IN KILOTONS (WHYIELD)	

WHPARA.DAT. This file contains the data for the weapons used by the fighter interceptors and the surface to air missiles against the pentrators (e.g. air-to-air missiles, guns. and SAM's).

PARAMETERS	VALUES

NUMBER OF FI WARHEAD TYPES
(FI_NUM_WH) INTEGER NO.
E.G. AIM7, AIM9, GUNS = 3.

NUMBER OF MI (SAM) WARHEADS
(MI_NUM_WH) INTEGER NO.
E.G. HAWK MISSILE = 1.

MATRIX OF WARHEAD PK's
ROWS = WH TYPES,
COLUMNS = PENETRATOR TYPES
(PK_SS)

	PEN#1	PEN#2 ...
FI WH 1		
FI WH 2		
FI WH 3		
MI WH 1		
etc		

NO. OF WARHEADS PER SALVO
FOR EACH WH TYPE, INTEGER
(SET TO ONE IN THESIS)
(SHOOT)

MAXIMUM LAUNCH RANGE FOR EACH
WARHEAD/MISSILE TYPE, METERS
*SET TO 0 FOR MI WH's
(RLAUNCH)

MINIMUM ASPECT ANGLE FOR FI WH's
*INPUT FOR AIR TO AIR MISSILES ONLY
UNITS = RADIANS (ASPMIN)

MAXIMUM ASPECT ANGLE FOR FI WH's
(ASPMAX)

WARHEAD SPEEDS IN METERS/SEC
FOR ALL WH TYPES (VWHEAD)

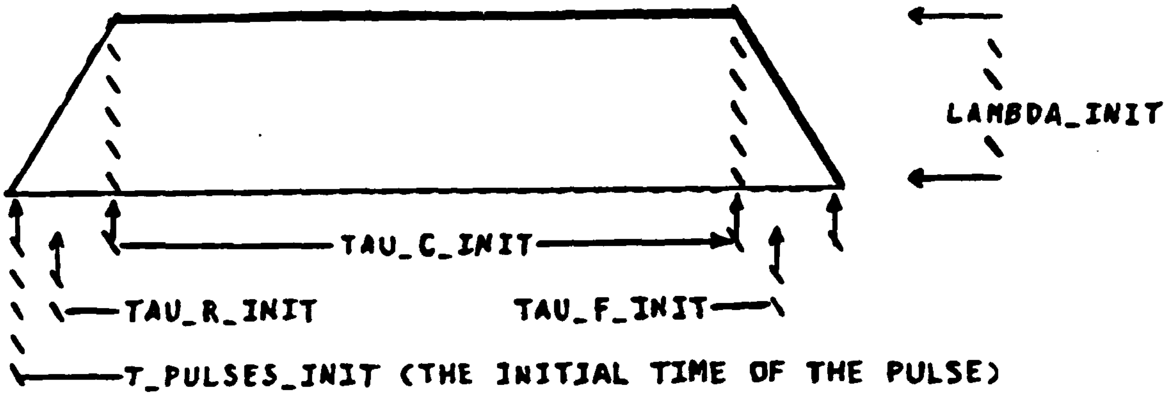
MATRIX OF FIGHTER INTERCEPTOR WH LOADS
ROW = WH TYPES, COL = FI TYPES
(INITWH)

	FI TYPE 1	FI TYPE 2
AIM7		
AIM9		
GUNS		

BASES.DAT. This file simply lists the fighter interceptor bases and their locations. There is also a parameter for fighter service times. Appendix C has an example of this file.

PARAMETERS	VALUES
FIGHTER "TURN" TIME AT THE BASE UNITS ARE SECONDS (BASESERVICETIME)	
NUMBER OF AIRBASES, MAX OF 10 RECOMMEND ONE BASE PER CAP (NAIRBASES)	
LOCATION OF AIRBASES LAT, LONG, ALTITUDE TENTHS OF DEGREES AND METERS ONE LINE PER BASE (BASELIST)	

PENTIM.DAT. This file contains parameters that TADZ uses to generate penetrator arrival times. The parameters describe "pulses" along a penetrator path. Each pulse is characterized by the five parameters listed below. This file is perhaps the most difficult to understand. Refer to the TADZ User's Maunual for further details [Merriman et al, 1984:3.54]. An example is supplied in Appendix C.



WHERE:

LAMBDA_INIT = MAXIMUM ARRIVAL RATE OF THE PENETRATORS IN THE PULSE (NO. OF PENETRATORS PER SECOND). LAMBDA_INIT > 0.

T_PULSES_INIT = TIME IN SECONDS AT WHICH EACH PULSE BEGINS. MUST BE GREATER THAN OR EQUAL TO ZERO.

TAU_R_INIT = THE LENGTH OF TIME OVER WHICH THE ARRIVAL RATE OF THE PENETRATORS INCREASES (SECONDS). MUST BE GREATER THAN OR EQUAL TO ZERO.

TAU_C_INIT = THE LENGTH OF TIME OVER WHICH THE ARRIVAL RATE OF THE PENETRATORS REMAINS CONSTANT (SECONDS). MUST BE GREATER THAN OR EQUAL TO ZERO.

TAU_F_INIT = THE LENGTH OF TIME OVER WHICH THE ARRIVAL RATE OF THE PENETRATORS DECREASES (SECONDS). MUST BE GREATER THAN OR EQUAL TO ZERO.

** NOTE: THE NUMBER OF PENETRATORS FOUND IN A GIVEN PULSE IS DEFINED BY THE AREA OF THE PULSE AS SHOWN ABOVE, I.E.

AREA = LAMBDA_INIT * (TAU_R_INIT/2 + TAU_C_INIT + TAU_F_INIT/2)

EXAMPLE: DEFINE ONE PENETRATOR PER PULSE....

LET   TAU_R_INIT = 0; TAU_F_INIT = 0; TAU_C_INIT = 1;
       LAMBDA_INIT = 1.

THE AREA OR NUMBER OF PENETRATORS IN THE PULSE WILL BE....

1*(0/2 + 1 + 0/2) = 1

====> A SINGLE PENETRATOR WILL ARRIVE BETWEEN T
T_PULSES_INIT AND T_PULSES_INIT + 1 SECONDS

PARAMETER	VALUE
NUMBER OF PATHS USED BY PENETRATOR CANNOT EXCEED NUMBER IN SYSLIM.DAT (N_RAID_PATHS)	

*** FOR EACH PATH YOU DESIRE TO "SEND" PENETRATORS DOWN, THE FOLLOWING PARAMETERS MUST BE DEFINED...

THE SPECIFIC PATH USED IN THIS PULSE
RELATIVE POSITION IN PATHSI.DAT
(RAIDPATHS)

NUMBER OF DIFFERENT PENETRATOR TYPES
USED ON THIS PATH (NPTYPES)

** FOR EACH PENETRATOR TYPE USED ON THIS PATH, DEFINE THE FOLLOWING PARAMETERS...

PENETRATOR TYPE USED
(PTYPES)

NUMBER OF ARRIVAL PULSES FOR
EACH PENETRATOR TYPE (NPULSEINIT)

```
* FOR EACH PULSE DEFINE

   PARAMETERS FOR EACH
   INDIVIDUAL PULSE (PULSE)

    LAMBDA_INIT

    TAU_R_INIT

    TAU_C_INIT

    TAU_F_INIT

    T_PULSES_INIT
```

NOTE: THIS FILE ESSENTIALLY CONSISTS OF TRIPLE NESTED LOOPS....

```
I = 1 TO NUMBER OF RAID PATHS
    J = 1 TO NO. OF PENETRATOR TYPES ON THE PATH
        K = 1 TO NO. PULSES PER PEN. TYPE ON THE PATH
```

COxx.DAT. This file type contains the information that describes the C^O's in the scenario. There can be two types of C^O's: the FI and MI or fighter and SAM types. Within the file or sequence of files the FI C^O's must be defined first. Some parameters in this file are specific to only one or the other of the types (MI or FI). These will be pointed out in the following parameter list.

PARAMETER	VALUE	
C^O TYPE DESIGNATION EITHER FI OR MI (COTYPE)		
C^O OPERATES AUTONOMOUSLY TRUE OR FALSE (COAUTO)		
LOCATION OF THE C^O LAT, LONG, ALTITUDE (COLOCATION)		
STRUCTURE DESIGNATION (COSTRUCTURE)	1	
NUMBER OF POINTS IN C^O BOUNDARY MAX OF 20 POINTS (COGBOUND)		
LAT. LONG OF BOUNDARY POINTS	LAT	LONG
MI C^O LETHAL REGION OF COVERAGE EXPRESSED AS PARABOLOID WITH XO = LAT. RANGE; YO = LONG. RANGE; ZO = ALTITUDE RANGE; ALL IN METERS (COFBOUND) *USE DUMMY VALUES FOR FI.		

NUMBER OF C^1 SUPERS FOR THIS C^0
(COC1)

INDEX OF C^1 SUPER (RELATIVE POSITION
IN EXEC FILE) LEAVE BLANK IF NO C^1 SUPER.
(ICOC1)

NO. OF C^2 SUPERS FOR THIS C^0
(COC2)

INDEX OF C^2 SUPER, LEAVE BLANK IF NONE
(ICOC2)

TIME TO PROCESS FIRST REPORT (FRSTCOSERVICE)

TIME TO PROCESS CONTINUED REPORTS
(CONTCOSERVICE)

TIME TO FUSE TWO MESSAGES ON SAME TARGET
(FUSNCOSERVICE)

LOITER POINT FOR FI C^0
LAT, LONG, ALT (LOITERPOINT)
* LEAVE BLANK IF MI C^0

BASE THAT SERVICES C^0 AIRCRAFT
RELATIVE POSITION IN BASES.DAT FILE
(BASENUMBER) LEAVE BLANK IF MI

NUMBER OF GCI OR FIRE CONTROLLERS
AVAILABLE FOR THIS C^0 (NUMOFSERVERS)

TOTAL NUMBER OF FI/MI TYPES...
FOR FI, SPECIFY TOTAL NO. OF FI TYPES
FOR MI, SPECIFY TOTAL FI & MI TYPES
(NUNITYPES)

THE FOLLOWING ENTRIES SPECIFY THE TYPES OF FI OR MI AT THIS C^{O}. THESE TYPES ARE DEFINED IN THE WEPARA.DAT FILE IN NUMERICAL SEQUENCE (I.E. TYPE 1,2,... N .LE. 4 ARE FI TYPES: WHILE 5,6,... N .LE. 14 ARE MI TYPES). EVEN THOUGH A MI C^{O} DOES NOT HAVE ANY FI's AND POSSIBLY NOT CERTAIN TYPES OF MI's, TADZ REQUIRES AN ENTRY FOR EACH WEAPON TYPE (I.E. J = 1,2,....N). PROVIDE DATA ACCORDING TO THE PARAMETER TEMPLATE FOR TYPE 1 (J=1) FOR ALL FI AND MI TYPES. FOR FI's, MI DATA NEED NOT BE INCLUDED SINCE IT COMES AFTER THE FI's IN THE SEQUENCE AS SEEN IN WEPARA.DAT.

START A LOOP: J = 1 TO NUUNITTYPES....

NUMBER OF UNITS BY TYPE (NUMUNITS)
FOR FI THE NO. OF FLIGHTS
FOR MI THE NUMBER OF LAUNCHERS
* SET = 0 IF THIS J (MI OR FI TYPE)
DOES NOT EXIST AT THIS C^{O}

INDEX OF UNIT TYPE CORRESPONDING
TO POITION IN WEPARA.DAT (COUNITS)

NUMBER OF WEAPONS PER UNIT (INITWEAPONS)
FOR FI's. NO. OF AIRCRAFT PER FLIGHT
E.G. 2,2,2
FOR MI's, NO. OF MISSILES PER LAUNCHER
E.G. 3,3,3,3

UNIT CONTROL POLICIES (INITCONT)
ONLY REQUIRED FOR FI's
SPECIFY FOR EACH FLIGHT
E.G. TITE,TITE,LOOS,LOOS...

UNIT STATUS (INITUNITSTAT)
FOR FI's, SPECIFY FOR EACH FLIGHT
E.G. LOIT,LOIT
* ONLY LOIT IS ENABLED IN CURRENT TADZ
FOR MI's, STATUS OF EACH LAUNCHER
E.G. READ,READ,READ

Executive File. This file "tells" TADZ where to look for the required input data. TEST.DAT is the file name used in this tactical version. Within TEST.DAT, the files are simply listed in a logical order along with some documentation concerning the purpose of the various input files. This is a simple but important input file. Refer to Appendix C for an example.

RADARx.DAT. The technical specifications for each radar type used in the simulation are included in this file. For this scenario, only two different types are used: the TPS43 EW radar and the HAWK acquisition radar. The variable NUM_RADAR_TYPE must be included as the first input in the first radar file read in, but is not included in subsequent radar files. In this case, NUM_RADAR_TYPE is set equal to 2 since there are two radar types modeled in the scenario. Consequently, there should be two radar files created (RADAR1.DAT and RADAR2.DAT for example).

RADARTAB.DAT. This file contains the radar pulse integration table. This table contains the signal-to-noise ratio (dB) needed to allow for a 50% chance of detecting a target given a certain number of integrated pulses. This figure is used in the calculation of radar burnthrough range. Appendix C contains working examples of the two radar files.

The documentation should be self explainatory. The radar parameters are important in that they are used to compute the coverage range of the radars.

APPENDIX C: TADZ INPUT FILES

The listing of several of the major input files for the TADZ model are contained in this appendix. These files were generated using the examples from Appendix B. The files are provided as samples of the data used to represent the fighter and missile units in the TADZ model. Each file starts on a new page except for the PKBLK.DAT file. The files and a brief explanation of each file included are listed below. For a detailed description of all the necessary files see the TADZ users guide provided by with the TADZ model.

CASE1.DAT. (page C-3) This is the file that calls all the other data files that are to be included in the TADZ model. The order of the files is important both within and without the blocks.

SYSLIM1.DAT. (page C-7) This file sets the limits of the TADZ execution run. It determines the number of nodes (radars, fighter C^2s and missile C^2s, etc.) to be included in the scenario model.

RADAR5.DAT. (page C-8) The data in this file is derived from unclassified sources and describes the parameters of the surveillance radar.

R01GSI1.DAT. (page C-8) This file defines the location of the surveillance radar and establishes the reporting structure, based on the order of the files listed in DAEB1.DAT.

S101.DAT. (page C-10) This is the movements and identification section (DIS/IBS) node. All S^{O}s report to this node.

C01FI1.DAT and C04MI1.DAT. (pages C-11 and C-14) These files determine the location and the number of fighters and missiles available to the model.

PATHBI.DAT. (page C-17) This file sets up the penetrator paths to be used during the model. The turn points and altitudes of the penetrators are set in this file.

PENSI.DAT. (page C-20) The radar cross sections for each type of penetrator are set in this file.

PENTIM.DAT. (page C-21) This file sets the number and timing of the penetrators that will enter the air defense zone. The raid can be set by penetrator type, time and path.

PKBLK.DAT. (page C-49) The probability of a penetrator to destroy an air defense fighter is set in this file. The probability is only used in a short range engagement.

WEPARA.DAT. (page C-49) The weapons (fighter and missile) parameter characteristics are set in this file and determine the specific capabilities (fuel, turn rates, speeds, etc.) of each fighter and SAM site.

```
H
H       THIS IS THE EXECUTIVE DATA SET FOR THE TADZ SIMULATION.
H       CONTAINED IN THIS FILE IS ALL STATIC CONTROL DATA LOADED FROM
H       DISK
H
B       SYSLIM
B
B       SPECIFICATIONS ON THE VARIOUS NUMBERS OF SYSTEM TYPE ELEMENTS
B       IN THE C3 SYSTEM
B
        'TEST:SYSLIM1.DAT'
*
B       RADAR
B
B       RADAR SET SPECIFICATION DATA
B
        'TEST:RADAR1.DAT'
      'TEST:RADAR2.DAT'
      'TEST:RADAR3.DAT'
      'TEST:RADAR4.DAT'
      'TEST:RADAR5.DAT'
      'TEST:RADAR6.DAT'
        'TEST:RADARTAB.DAT'
*
B       R0INIT
B
B       INITIAL RADAR SPECIFICATIONS
B
        'TEST:R01SCI1.DAT'
      'TEST:R02SCI2.DAT'
      'TEST:R03SCI3.DAT'
        'TEST:R04SAM1.DAT'
      'TEST:R05SAM2.DAT'
      'TEST:R06SAM3.DAT'
      'TEST:R07SAM4.DAT'
*
B       S0INIT
B
B       INITIAL S0 SPECIFICATIONS
B
        'TEST:S01SCI1.DAT'
      'TEST:S02SCI2.DAT'
      'TEST:S03SCI3.DAT'
      'TEST:S04SAM1.DAT'
      'TEST:S05SAM2.DAT'
      'TEST:S06SAM3.DAT'
      'TEST:S07SAM4.DAT'
*
B       S1INIT
B
B       INITIAL S1 SPECIFICATIONS
```

```
B
        'TEST:S101.DAT'
$
B       S2INIT
B
B       INITIAL S2 SPECIFICATIONS
B
        'TEST:S201.DAT'
$
B       C2INIT
B
B       INITIAL C2 SPECIFICATIONS
B
        'TEST:C201.DAT'
$
B       C1INIT
B
B       INITIAL C1 SPECIFICATIONS
B
        'TEST:C101.DAT'
$
B       COINIT
B
B
B       INITIAL CO ASSETS
B
B       THE DATA FILES FOR THE CO ASSETS
B
        'TEST:CO1FI1.DAT'
        'TEST:CO2FI2.DAT'
       'TEST:CO3FI3.DAT'
       'TEST:CO4MI1.DAT'
       'TEST:CO5MI2.DAT'
       'TEST:CO6MI3.DAT'
       'TEST:CO7MI4.DAT'
$
B       WEPARA
B
B       SYSTEM WEAPON PARAMETERS
B
        'TEST:WEPARA.DAT'
$
B       WHPARA
B
B       SYSTEM WARHEAD PARAMETERS
B
        'TEST:WHPARA.DAT'
$
B       COPARA
B
B       CO PARAMETERS
B
```

```
        'TEST:COPARA.DAT'
*
B       BASES
B
B       LOCATIONS OF SYSTEM AIR BASES
B
        'TEST:BASES.DAT'
*
B       PATHSI
B
B       PENETRATION PATH DATA
B
        'TEST:PATHSI.DAT'
*
B       PENSI
B
B       DESCRIPTION OF THE PENETRATOR TYPES
B
        'TEST:PENSI.DAT'
*
B       PENTIM
B
B       PENETRATION RAID GENERATION DATA
B
        'TEST:PENTIM.DAT'
*
B       TARGET
B
B       SPECIFICATIONS FOR PENETRATOR TARGET STRIKES
B
        'TEST:NOTARGETS.DAT'
*
B       WRHEAD
B
B       CAPABILITIES OF THE PENETRATOR WARHEADS
B
        'TEST:WRHEAD.DAT'
*
B     COORD
B
B     THE COORDINATE INFORMATION BLOCK
B
      'TEST:COORD.DAT'
*
B       PKBLK
B
B       PK'S FOR PENETRATORS AGAINST UNITS
B
        'TEST:PKBLK.DAT'
*
B       COPRI
B
```

```
B      ORDERING OF SERVICES FOR THE C3 QUEUES
B
       'TEST:C3PRI.DAT'
*
B      STRUCT
B
B      HARDNESS TYPES OF C3 SYSTEM ELEMENTS
B
       'TEST:STRUCT.DAT'
*
B      RUNCON
B
B      EXECUTION CONTROL DATA
B
       'TEST:RUNCON.DAT'
*
B      SLMCON
B
B      CONTROL DATA FOR SLAM
B
       'TEST:SLMCON.DAT'
*
B      STTCON
B
B      CONTROL OF STATISTICS COLLECTION. AND NUMBER OF REPLICATIONS
B
       'TEST:STTCON.DAT'
*
B      AGGFLS
B
B      AGGREGATE STATISTICS OPTIONS
B
       'TEST:AGGFLS.DAT'
*
B      SEEDS
B
B      INITIAL VALUES OF THE RANDOM NUMBER SEEDS
B
       'TEST:SEEDS1.DAT'
*
B      CMINPT
B
B      MESSAGE DELAYS (PERHAPS GENERATED BY THE COMMUNICATION MODELS)
B
       'TEST:CMINPT2.DAT'
*
B     SWITCH
B
B     CENTRALIZED OF DECENTRALIZED EXECUTION
B
      'TEST:SWITCHES.DAT'
*
```

```
R      SYSLIM
V      INTEGER NUMRADAR----------NUMBER OF RADARS IN THE C3 SYSTEM
F      NUMRADAR
       7
V      INTEGER NUMS0-------------NUMBER OF S0'S IN THE C3 SYSTEM
F      NUMS0
       7
V      INTEGER NUMS1-------------NUMBER OF S1'S IN THE C3 SYSTEM
F      NUMS1
       1
V      INTEGER NUMS2-------------NUMBER OF S2'S IN THE C3 SYSTEM
F      NUMS2
       1
V      INTEGER NUMC2-------------NUMBER OF C2'S IN THE C3 SYSTEM
F      NUMC2
       1
V      INTEGER NUMC1-------------NUMBER OF C1'S IN THE C3 SYSTEM
F      NUMC1
       1
V      INTEGER NUMC0-------------NUMBER OF C0'S IN THE C3 SYSTEM
F      NUMC0
       7
V      INTEGER NUM_PEN_TYPES-----NUMBER OF PENETRATOR TYPES ACTUALLY
V                               -----USED IN SCENARIO
F      NUMPENTYPES
       2
V      INTEGER NUMPATHS----------THE NUMBER OF PATHS USED IN THIS
V                                      SCENARIO
F      NUMPATHS
       8
```

```
R     RADARSEQ
V
V     FOR I=1.NUM_RADAR_TYPE SPECIFY THE FOLLOWING
V
V     I=3
V
V     REAL      RADAR_FREQUENCY(I)-----RADAR CENTER FREQUENCY (MHZ)
F     RFREQUENCY
      3000
V     REAL      RADAR_POWER(I)---RADAR TRANSMISSION POWER  (KW)
F     RPOWER
      6.7
V     REAL      RADAR_PRF(I)----RADAR PULSE REPETITION FREQUENCY  (PPS)
F     RPRF
      258
V     REAL      RADAR_PULSE_WIDTH(I)-----RADAR PULSE WIDTH (MICRO-S)
F     RPULSEWIDTH
      6.5
V     REAL      RADAR_SWEEP_RATE (I)-----RADAR ANTENNA SWEEP RATE (RPM)
F     RSWEEPRATE
      6
V     REAL      RADAR_BEAMWIDTH(I)-------RADAR ANGULAR BEAMWIDTH (DEG)
F     RBEAMWIDTH
      1.1
V     REAL      RADAR_ANTENNA_GAIN(I)----MAXIMUM RADAR ANTENNA GAIN (DB)
F     RANTENNAGAIN
      36
V     REAL      RADAR_MAX_ALT(I)---------RADAR MAX HEIGHT FINDING ALTITUDE (KM)
F     RMAXALT
      20
V     REAL      RADAR_TRANS_LOSS(I)------RADAR TRANSMISSION LOSS FACTOR (ND)
F     RTRANSLOSS
      4
V     REAL      RADAR_REC_LOSS(I)--------RADAR RECEPTION LOSS FACTOR(ND)
F     RRECLOSS
      1
V     RADAR_TYPE--------------EARLY WARNING (EW) OR ACQUISITION  (ACQ)
F     RTYPE
      EW
V     INTEGER RADAR_REPORT_CYCLE-------NUMBER OF SWEEPS BETWEEN REPORTS
F     RREPORTCYCLE
      1
```

```
R     ROINIT
V
V     RADAR SPECS
V
V     I=1
V
V     REAL      RO_CP_LAT(I),RO_CP_LONG(I).RO_CP_ALT(I)
F     ROLOCATION
      100E3, 162.5E3, 0
V     INTEGER RO_RADAR_SET(I)--------------RADAR SET TYPE FOR THE RO SITE
F     ROTYPE
      5
V
V               INTEGER N_MASK_ANGLE(I)--------------THE NUMBER OF MASKING ANGLES AT
V                                   --------------A RADAR SITE
F     NUMMASKANGLES
      4
V
V     FOR J=1, N_MASK_ANGLE(I), SPECIFY THE FOLLOWING
V
V     REAL      PSI_MASK(I,J),EPS_MASK(I,J)--THE REFERENCE AZIMUTHS, ELEVATIONS
V                                              FOR MASKING AT A RADAR SITE
F     MASKANGLES
      0,0
      90.0
      180,0
      270.0
V     INTEGER RO_STRUCTURE(I)----STRUCTURE TYPE HOUSING THE CP
F     ROSTRUCTURE
      1
V     INTEGER NO_RO_SO(I)----------------NUMBER OF SO SUPERVISORS FOR RO(I)
V       CAN ONLY HAVE ONE SUPERVISOR
F     NOROSO
      1
V     INTEGER I_RO_SO(I,1)------------INDEX OF THE SO SUPER FOR THIS RO(I)
F     IROSO
      1
V     REAL      FIRST_RO_SERVICE(I)--------FIRST RO SERVICE TIME
F     FRSTROSERVICE
      0.0
V     REAL      CONT_RO_SERVICE(I)---------CONTINUED RO SERVICE TIME
F     CONTROSERVICE
      0.0
V     REAL      FUSION_RO_SERVICE(I)-------FUSION RO SERVICE TIME
F     FUSNROSERVICE
      0.0
R     S1INIT
V
V
V     THIS BLOCK CONTAINS THE INITIAL S1 SPECIFICATIONS
V^C
```

```
R     S1INIT
V
V
V     THIS BLOCK CONTAINS THE INITIAL S1 SPECIFICATIONS
V
V     FOR I=1.NUMS1 SPECIFY THE FOLLOWINS
V
V     I=1
V
V     REAL      S1_CP_LAT(I),S1_CP_LONG(I).S1_CP_ALT(I)
F     S1LOCATION
      100E3. 100E3. 0
V     INTEGER   S1_STRUCTURE(I)-------STRUCTURE DESIGNATION
F     S1STRUCTURE
      1
V     INTEGER NO_S1_S2(I)--------------NUMBER OF S2 SUPERVISORS FOR S1(I)
R     S1S2
F     NOS1S2
      1
V     INTEGER I_S1_S2(I,1)-------------INITIAL S2 SUPERVISORS FOR S1(I)
F     IS1S2
      1
V     INTEGER NO_S1_C1(I)--------------NUMBER OF C1 SUPERVISORS FOR S1(I)
R     S1C1
F     NOS1C1
      0
V     INTEGER I_S1_C1(I,1)-------------INITIAL C1 SUPERVISORS FOR S1(I)
F     IS1C1
V
V     REAL      FIRST_S1_SERVICE(I)--------SERVICE TIME 1 FOR S1
R     S1INIT
F     FRSTS1SERVICE
      1.0
V     REAL      CONT_S1_SERVICE(I)---------SERVICE TIME 2 FOR S1
F     CONTS1SERVICE
      0
V     REAL      FUSION_S1_SERVICE(I)-------SERVICE TIME 3 FOR S1
F     FUSNS1SERVICE
      0
V     INTEGER S1_S2_SELECT(I)------------SUBSTITUTE PRIORITY
F     S1S2SELECT
      0
V     REAL      S1_S2_MULTIPLIER(I)--------SUBSTITUTE MULTIPLIER
F     S1S2MULTIPLIER
      1
```

```
R     COINIT
V
V     FOR I=1.NUMCO PROVIDE THE FOLLOWING
V
V     I=1
V
V     INTEGER   COTYP(I)-------------CO TYPE DESIGNATION (FI OR MI)
F     COTYPE
      FI
V     LOGICAL CO_AUTO(I)-------------CO AUTO CLASS IDENTIFIER (TRUE FOR AUTO)
V       NO AUTONOMOUS OPERATION ALLOWED FOR CAPS
F     COAUTO
      F
V     REAL      CO_CP_LAT(I),CO_CP_LONG(I),CO_CP_ALT(I)
V     IN METERS!!!!!!!
F     COLOCATION
        90E3. 165E3, 0
V     INTEGER   CO_STRUCTURE(I)-------STRUCTURE DESIGNATION
F     COSTRUCTURE
      1
V     INTEGER CO_G_BOUND(I)--------NUMBER OF GRID POINTS FORMING
V                                --------THE BOUNDARY OF CO
F     COGBOUND
      4
V     REAL      CO_LAT_BOUND(I,J),CO_LONG_BOUND(I,J),J=1,CO_G_BOUND(I)
V               ---------LATITUDE AND LONGITUDE POINTS OF THE BOUNDARY
V     IN METERS!!!!!!!
F     COBOUNDARY
      15.     140E3
      121E3.  130E3
      110E3.  199E3
      15.     199E3
V     REAL      CO_F_BOUND(I.3)------X0.Y0.Z0 SPECIFICATIONS FOR A PARABOLOID
R     COINIT
F     COFBOUND
      0.0.0
V     INTEGER NO_CO_C1(I)----------NUMBER OF C1 SUPERVISORS FOR CO(I)
R     COC1
F     NOCOC1
      0
V     INTEGER I_CO_C1(I.1)---------INITIAL C1 SUPERVISORS FOR CO(I)
V       INDEX OF C1 SUPERVISOR
F     ICOC1
V     INTEGER NO_CO_C2(I)----------NUMBER OF C2 SUPERVISORS FOR CO(I)
R     COC2
F     NOCOC2
      1
V     INTEGER I_CO_C2(I.1)---------INITIAL C2 SUPERVISORS FOR CO(I)
F     ICOC2
      1
V     REAL      FIRST_CO_SERVICE(I)--------SERVICE TIME 1 FOR CO
R     COINIT
```

```
F     FRSTCOSERVICE
      60
V     REAL      CONT_CO_SERVICE(I)---------SERVICE TIME 2 FOR CO
F     CONTCOSERVICE
      0.0
V     REAL      FUSION_CO_SERVICE(I)--------SERVICE TIME 3 FOR CO
F     FUSNCOSERVICE
      0.0
V     FOR COTYP=FI. SPECIFY LOITER LAT.LONG.ALT. BASE NUMBER
V     REAL      CO_LOITER_LAT.CO_LOITER_LONG.CO_LOITER_ALT
F     LOITERPOINT
      80E3. 165E3. 5E3
V     INTEGER   BASE_NUMBER
F     BASENUMBER
      1
V     INTEGER   NUM_OF_SERVERS(I)----NUMBER OF CO SERVERS
F     NUMOFSERVERS
        8
V       NUMBER OF SIMULTANEOUS INTERCEPTS BY GCI CONTROLLER
V
V     INTEGER   N_UNIT_TYPES(I)------NUMBER OF UNIT TYPES
V       NUMBER OF DIFFERENT TYPES OF FI AS DEFINED IN WEPARA
F     NUNITYPES
      1
V
V     FOR EACH UNIT TYPE PROVIDE THE FOLLOWING FOR J=1,N_UNIT_TYPES
V       J IS THE INDEX FOR THE FIGHTER TYPE
V     J=1
V
V     INTEGER   N_INIT_UNITS(I,J)----INITIAL NUMBER OF UNITS BY TYPE
V       NUMBER OF FLIGHTS OF THIS TYPE
R     FIUNIT
F     NUMUNITS
      10
V     INTEGER   CO_UNIT_TYPES(I,J)---SPECIFIC CO UNIT TYPES
F     COUNITS
        1
V       THIS IS THE INDEX OF THE FI TYPE FROM THE WEPARA FILE
V
V     FOR K=1,N_INIT_UNITS(I,J) SPECIFY THE FOLLOWING:
V       K IS THE INDEX FOR THE TOTAL NO. OF FIGHTER TYPE J
V     INTEGER   INIT_WEAPONS_UNIT(I,J,K)----INITIAL NUMBER OF WEAPONS PER UNIT
V       NUMBER OF AIRCRAFT PER FLIGHT
F     INITWEAPONS
        2,2,2,2,2,2,2,2,2,2
V
V     FOR COTYP(I)=FI SPECIFY THE FOLLOWING:
V
V     INTEGER   INIT_CONT(I,J,K)------------INITIAL UNIT CONTROL POLICIES
V                                 ------------FOUR LETTER LITERALS
V                               ------------FORMATTED:  A001,A002,...,A010
V                                 ------------10 MAXIMUM PER LINE
```

```
F     INITCONT
      LOOS.LOOS.LOOS.LOOS.LOOS.LOOS.LOOS.LOOS,LOOS.LOOS
V
V       ONE ENTRY REQUIRED FOR EACH FLIGHT OF FI(J)
V
V     INTEGER   INIT_UNIT_STAT(I,J,K)-------INITIAL UNIT STATUS
V                                              ('READ' , 'LOIT' , 'BASE')
F     INITUNITSTAT
        LOIT.LOIT.LOIT,LOIT.LOIT.LOIT.LOIT,LOIT,LOIT,LOIT
V
V     END OF FILE
```

```
R     ^C
R     COINIT
V
V     FOR I=1,NUMCO PROVIDE THE FOLLOWING
V
V     I=4
V     INTEGER   COTYP(I)-------------CO TYPE DESIGNATION (FI OR MI)
F     COTYPE
      MI
V     LOGICAL CO_AUTO(I)-------------CO AUTO CLASS IDENTIFIER (TRUE FOR AUTO)
V       AUTONOMOUS OPERATION ALLOWED (TRUE/FALSE)
F     COAUTO
      F
V     REAL       CO_CP_LAT(I).CO_CP_LONG(I),CO_CP_ALT(I)
V     IN METERS!!!!!!!
F     COLOCATION
      160E3, 165E3. 0
V     INTEGER   CO_STRUCTURE(I)-------STRUCTURE DESIGNATION
F     COSTRUCTURE
      1
V     INTEGER CO_G_BOUND(I)--------NUMBER OF GRID POINTS FORMING
V                                 --------THE BOUNDARY OF CO
F     COGBOUND
      4
V     REAL       CO_LAT_BOUND(I.J),CO_LONG_BOUND(I.J),J=1,CO_G_BOUND(I)
V               ---------LATITUDE AND LONGITUDE POINTS OF THE BOUNDARY
V     IN METERS!!!!!!!
F     COBOUNDARY
      120E3.145E3
      199E3,155E3
      199E3.199E3
      110E3,199E3
V     REAL       CO_F_BOUND(I.3)------X0,Y0.Z0 SPECIFICATIONS FOR A PARABOLOID
R     COINIT
F     COFBOUND
      30E3,30E3,100
V     INTEGER NO_CO_C1(I)----------NUMBER OF C1 SUPERVISORS FOR CO(I)
V       CAN ONLY HAVE ONE SUPERVISOR
R     COC1
F     NOCOC1
      1
V     INTEGER I_CO_C1(I,1)---------INITIAL C1 SUPERVISORS FOR CO(I)
V       INDEX OF ACTUAL C1 SUPERVISOR
F     ICOC1
      1
V     INTEGER NO_CO_C2(I)----------NUMBER OF C2 SUPERVISORS FOR CO(I)
R     COC2
F     NOCOC2
      0
V     INTEGER I_CO_C2(I.1)---------INITIAL C2 SUPERVISORS FOR CO(I)
V       CAN ONLY HAVE A C1 OR A C2 SUPERVISOR
F     ICOC2
```

```
V     REAL      FIRST_CO_SERVICE(I)--------SERVICE TIME 1 FOR CO
R     COINIT
F     FRSTCOSERVICE
      0.0
V     REAL      CONT_CO_SERVICE(I)---------SERVICE TIME 2 FOR CO
F     CONTCOSERVICE
      0.0
V     REAL      FUSION_CO_SERVICE(I)-------SERVICE TIME 3 FOR CO
F     FUSNCOSERVICE
      0.0
V     FOR COTYP=FI. SPECIFY LOITER LAT,LONG,ALT, BASE NUMBER
V     REAL      CO_LOITER_LAT,CO_LOITER_LONG,CO_LOITER_ALT
F     LOITERPOINT
V     INTEGER   BASE_NUMBER
F     BASENUMBER
V     INTEGER   NUM_OF_SERVERS(I)----NUMBER OF CO SERVERS
V       NUMBER OF FIRE CONTROLLERS AVAILABLE
F     NUMOFSERVERS
      3
V     INTEGER   N_UNIT_TYPES(I)------NUMBER OF UNIT TYPES
F     NUNITYPES
      2
V
V     FOR EACH UNIT TYPE PROVIDE THE FOLLOWING FOR J=1,N_UNIT_TYPES
V
V     J=1
V
V     INTEGER   N_INIT_UNITS(I,J)----INITIAL NUMBER OF UNITS BY TYPE
R     FISTUFF
F     NUMUNITS
      0
V     INDEX OF CO UNIT TYPE
F     COUNITS
      1
F       INITWEAPONS
V
F       INITCONT
V
F       INITUNITSTAT
V
V     J=2
V
V     INTEGER N_INIT_UNITS(I,J)-----INITIAL NUMBER OF UNITS BY TYPE
F     NUMUNITS
      6
V
V     INTEGER   CO_UNIT_TYPES(I,J)---SPECIFIC CO UNIT TYPES
F     COUNITS
        2
V       THIS IS THE INDEX CORRESPONDING TO THE WEPARA FILE
V
V     FOR K=1,N_INIT_UNITS(I,J) SPECIFY THE FOLLOWING:
```

```
V
V     INTEGER   INIT_WEAPONS_UNIT(I,J,K)----INITIAL NUMBER OF WEAPONS PER UNIT
F     INITWEAPONS
        3,3,3,3,3,3
V
V     FOR COTYP(I)=FI SPECIFY THE FOLLOWING:
V
V     INTEGER   INIT_UNIT_STAT(I,J,K)-------INITIAL UNIT STATUS
V                                                ('READ' , 'LOIT' , 'BASE')
F     INITUNITSTAT
        READ,READ,READ,READ,READ,READ
V
```

```
R     PATHSI
V
V     PATHSI CONTAINS PATH DATA
V
V     FOR I=1,NPATHSI INPUT THE FOLLOWING
V
V     I=1
V
V     INTEGER N_POINTSI(N_PATH_MAX)-THE NUMBER OF POINTS ALONG EACH PATH
F     NPOINTSI
V
      3
V
V     FOR J=1,N_POINTSI SPECIFY THE FOLLOWING
V
V     REAL      PATH_LAT(I,J),PATH_LONG(I,J),PATH_ALT(I,J)
F     IPOINTSI
V
        499E3, 190E3, 90
        120E3, 190E3, 90
            5, 190E3, 90
V
V     I=2
V
V     INTEGER N_POINTSI(N_PATH_MAX)-THE NUMBER OF POINTS ALONG EACH PATH
F     NPOINTSI
V
      3
V
V     FOR J=1,N_POINTSI SPECIFY THE FOLLOWING
V
V     REA~BL    PATH_LAT(I,J),PATH_LONG(I,J),PATH_ALT(I,J)
F     IPOINTSI
V
        500E3, 145E3, 90
        95E3, 145E3, 90
        5    , 145E3, 90
V
V     I=3
V
V     INTEGER N_POINTSI(N_PATH_MAX)-THE NUMBER OF POINTS ALONG EACH PATH
F     NPOINTSI
V
      3
V
V     FOR J=1,N_POINTSI SPECIFY THE FOLLOWING
V
V     REAL      PATH_LAT(I,J),PATH_LONG(I,J),PATH_ALT(I,J)
F     IPOINTSI
      502E3, 137E3, 90
        105E3, 137E3, 90
        5    , 137E3, 90
```

```
V
V     I=4
V
V     INTEGER N_POINTSI(N_PATH_MAX)-THE NUMBER OF POINTS ALONG EACH PATH
F     NPOINTSI
V
      3
V
V     FOR J=1,N_POINTSI SPECIFY THE FOLLOWING
V
V     REAL      PATH_LAT(I,J),PATH_LONG(I,J),PATH_ALT(I.J)
F     IPOINTSI
      501E3. 105E3. 90
        110E3, 105E3. 90
        5    , 105E3, 90
V
V     I=5
V
V     INTEGER N_POINTSI(N_PATH_MAX)-THE NUMBER OF POINTS ALONG EACH PATH
F     NPOINTSI
V
      3
V
V     FOR J=1,N_POINTSI SPECIFY THE FOLLOWING
V
V     REAL      PATH_LAT(I,J),PATH_LONG(I,J),PATH_ALT(I,J)
F     IPOINTSI
      499E3, 95E3, 90
        115E3, 95E3, 90
        5    , 95E3, 90
V
V     I=6
V
V     INTEGER N_POINTSI(N_PATH_MAX)-THE NUMBER OF POINTS ALONG EACH PATH
F     NPOINTSI
V
      3
V
V     FOR J=1.N_POINTSI SPECIFY THE FOLLOWING
V
V     REAL      PATH_LAT(I.J).PATH_LONG(I.J).PATH_ALT(I,J)
F     IPOINTSI
      500E3. 60E3, 90
        120E3. 60E3. 90
        5    , 60E3, 90
V
V     I=7
V
V     INTEGER N_POINTSI(N_PATH_MAX)-THE NUMBER OF POINTS ALONG EACH PATH
F     NPOINTSI
V
      3
```

```
V
V     FOR J=1.N_POINTSI SPECIFY THE FOLLOWING
V
V     REAL       PATH_LAT(I,J),PATH_LONG(I,J),PATH_ALT(I,J)
F     IPOINTSI
      498E3. 53E3. 90
        130E3, 53E3, 90
        5    , 53E3, 90
V
V     I=8
V
V     INTEGER N_POINTSI(N_PATH_MAX)-THE NUMBER OF POI... ALONG EACH PATH
F     NPOINTSI
V
      3
V
V     FOR A J=1.N_POINTSI SPECIFY THE FOLLOWING
V
V     REAL       PATH_LAT(I,J),PATH_LONG(I,J),PATH_ALT(I,J)
F     IPOINTSI
      501E3, 10E3, 90
        100E3, 10E3, 90
        5    , 10E3, 90
V
V     END OF FILE
```

```
R     PENSI
V     FOR I=1.NUM_PEN_TYPES, SPECIFY THE FOLLOWING
V
V     I=1   BOMBER/FIGHTER BOMBER
V
V     INTEGER PEN_TYPES(I)-----------THE SPECIFIC PENETRATOR TYPES
F     PEN_TYPE
      1
V     REAL PEN_SPEED(I)-----------THE PENETRATOR SPEED
F     PENSPEED
      260
V     REAL PEN_CROSS(I)-----------THE PENETRATOR RADAR CROSS-SECTION
F     PENCROSS
      50
V     INTEGER PEN_BOMBS(I,5)---------THE NUMBER OF BOMBS OF EACH TYPE
V                                         CARRIED BY THE PENETRATORS
F     PENBOMBS
      4,8,0,0,0
V     REAL      ECM_POWER(I,2)---------THE ECM POWER FOR TWO TYPES OF
V                                         TRANSMITTERS (WATTS/MHZ)
F     ECMPOWER
      0,0
V
V     I=2   FIGHTER/FIGHTER ESCORT
V
V     INTEGER PEN_TYPES(I)-----------THE SPECIFIC PENETRATOR TYPES
F     PEN_TYPE
      2
V     REAL PEN_SPEED(I)-----------THE PENETRATOR SPEED
F     PENSPEED
      260
V     REAL PEN_CROSS(I)-----------THE PENETRATOR RADAR CROSS-SECTION
F     PENCROSS
      10
V     INTEGER PEN_BOMBS(I,5)---------THE NUMBER OF BOMBS OF EACH TYPE
V                                         CARRIED BY THE PENETRATORS
F     PENBOMBS
      4,8,0,0,0
V     REAL      ECM_POWER(I,2)---------THE ECM POWER FOR TWO TYPES OF
V                                         TRANSMITTERS (WATTS/MHZ)
F     ECMPOWER
      0,0
V
V
V     END OF FILE
```

```
V
V     BLOCK 'PENTIM' CONTAINS PENETRATOR ARRIVAL TIME INPUTS
V     INPUTS TO THE SYSTEM ARE DESCRIBED IN TERMS OF PULSES ALONG
V     A GIVEN PENETRATION PATH. EACH PULSE IS CHARACTERIZED BY
V     FIVE PARAMETERS, AS SHOWN BELOW.
V
V                 /-------------------------------\        <----
V                /¦                               ¦\           ¦
V               / ¦                               ¦ \          ¦
V              /  ¦                               ¦  \       LAMBDA_INIT
V             /   ¦                               ¦   \        ¦
V            /____¦_______________________________¦____\   <----
V            ^    ^                               ^    ^
V            ¦ ^  ¦-----------TAU_C_INIT----------¦ ^  ¦
V            ¦ ¦                                    ¦
V            ¦ ¦---TAU_R_INIT          TAU_F_INIT---¦
V            ¦
V            ¦-----T_PULSES_INIT (the initial time of the pulse)
V
V     THE NUMBER OF PENETRATORS FOUND IN A GIVEN PULSE WILL BE
V     LAMBDA_INIT * (TAU_R_INIT/2 + TAU_C_INIT + TAU_F_INIT/2).
V
V
V     INTEGER N_RAID_PATHS----------------THE NUMBER OF PATHS USED FOR RAIDS
F     N_RAID_PATHS
      8
V
V     FOR I=1,N_RAID_PATHS, SPECIFY THE FOLLOWING:
V
V     I=1
V
V     INTEGER RAID_PATHS(I)---------------THE PATH USED FOR THE RAID
R     PENTIMSEQ
F     RAIDPATHS
      1
V
V     INTEGER N_P_TYPES(I)----------------THE NUMBER OF PENETRATOR TYPES USED
F     NPTYPES
      1
V
R     PATHPEN
V     FOR J=1,N_P_TYPES, SPECIFY THE FOLLOWING:
F     PTYPES
V
V     J=1
V
V     INTEGER P_TYPES(I,J)----------------THE PENETRATOR TYPE USED
      1
V     INTEGER N_PULSE_INIT(I,J)-----------THE NUMBER OF ARRIVAL PULSES
F     NPULSEINIT
      10
V
```

```
V     FOR K=1,N_PULSE_INIT, SPECIFY THE FOLLOWING:
V
V     K=1
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      0
V
V     K=2
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      15
V
V     K=3
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      30
V
V
V     K=4
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
```

```
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      45
V
V
V     K=5
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL     TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      60
V
V
V     K=6
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL     TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      75
V
V
V     K=7
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL     TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      90
V
```

```
V
V     K=8
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      3
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      105
V
V
V     K=9
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      3
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      120
V
V
V     K=10
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      3
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      135
V
V
V
V     FOR I=1,N_RAID_PATHS, SPECIFY THE FOLLOWING:
V
V     I=2
V
V     INTEGER RAID_PATHS(I)--------------THE PATH USED FOR THE RAID
```

```
R       PENTIMSEQ
F       RAIDPATHS
        2
V
V       INTEGER N_P_TYPES(I)----------------THE NUMBER OF PENETRATOR TYPES USED
F       NPTYPES
        1
V
R       PATHPEN
V       FOR J=1,N_P_TYPES, SPECIFY THE FOLLOWING:
F       PTYPES
V
V       J=1
V
V       INTEGER P_TYPES(I,J)----------------THE PENETRATOR TYPE USED
        2
V       INTEGER N_PULSE_INIT(I,J)-----------THE NUMBER OF ARRIVAL PULSES
F       NPULSEINIT
        10
V
V       FOR K=1,N_PULSE_INIT, SPECIFY THE FOLLOWING:
V
V       K=1
V
F       PULSE
V       REAL        LAMBDA_INIT(I,J,K)----------THE DISTRIBUTION FUNCTION VALUE
        0.5
V       REAL        TAU_R_INIT(I,J,K)-----------THE FIRST PULSE PHASE VALUE
        0
V       REAL        TAU_C_INIT(I,J,K)-----------THE SECOND PULSE PHASE VALUE
        8
V       REAL        TAU_F_INIT(I,J,K)-----------THE LAST PULSE PHASE VALUE
        0
V       REAL        T_PULSES_INIT(I,J,K)--------THE START TIME OF THE PULSE
        0
V
V       K=2
V
F       PULSE
V       REAL        LAMBDA_INIT(I,J,K)----------THE DISTRIBUTION FUNCTION VALUE
        0.5
V       REAL        TAU_R_INIT(I,J,K)-----------THE FIRST PULSE PHASE VALUE
        0
V       REAL        TAU_C_INIT(I,J,K)-----------THE SECOND PULSE PHASE VALUE
        8
V       REAL        TAU_F_INIT(I,J,K)-----------THE LAST PULSE PHASE VALUE
        0
V       REAL        T_PULSES_INIT(I,J,K)--------THE START TIME OF THE PULSE
        15
V
V       K=3
V
```

```
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      30
V
V
V     K=4
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      45
V
V
V     K=5
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      60
V
V
V     K=6
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
```

```
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      75
V
V
V     K=7
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      90
V
V
V     K=8
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      105
V
V
V     K=9
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      120
V
V
V     K=10
```

```
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      135
V
V
V     FOR I=1,N_RAID_PATHS, SPECIFY THE FOLLOWING:
V
V     I=3
V
V     INTEGER RAID_PATHS(I)--------------THE PATH USED FOR THE RAID
R     PENTIMSEQ
F     RAIDPATHS
      3
V
V     INTEGER N_P_TYPES(I)---------------THE NUMBER OF PENETRATOR TYPES USED
F     NPTYPES
      1
V
R     PATHPEN
V     FOR J=1,N_P_TYPES, SPECIFY THE FOLLOWING:
F     PTYPES
V
V     J=1
V
V     INTEGER P_TYPES(I,J)---------------THE PENETRATOR TYPE USED
      1
V     INTEGER N_PULSE_INIT(I,J)----------THE NUMBER OF ARRIVAL PULSES
F     NPULSEINIT
      10
V
V     FOR K=1,N_PULSE_INIT, SPECIFY THE FOLLOWING:
V
V     K=1
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
```

```
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      0
V
V
V     K=2
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL     TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      15
V
V     K=3
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL     TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      30
V
V
V     K=4
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL     TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      45
V
V
V     K=5
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
```

```
        0.5
V       REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
        0
V       REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
        8
V       REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
        0
V       REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
        60
V
V
        K=6

F       PULSE
V       REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
        0.5
V       REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
        0
V       REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
        8
V       REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
        0
V       REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
        75
V
V
V       K=7
V
F       PULSE
V       REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
        0.5
V       REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
        0
V       REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
        8
V       REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
        0
V       REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
        90
V
V
V       K=8
V
F       PULSE
V       REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
        0.5
V       REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
        0
V       REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
        8
V       REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
        0
```

```
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      105
V
V
V     K=9
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      120
V
V
V     K=10
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      135
V
V
V
V     FOR I=1,N_RAID_PATHS, SPECIFY THE FOLLOWING:
V
V     I=4
V
V     INTEGER RAID_PATHS(I)---------------THE PATH USED FOR THE RAID
R     PENTIMSEQ
F     RAIDPATHS
      4
V
V     INTEGER N_P_TYPES(I)----------------THE NUMBER OF PENETRATOR TYPES USED
F     NPTYPES
      1
V
R     PATHPEN
V     FOR J=1,N_P_TYPES, SPECIFY THE FOLLOWING:
F     PTYPES
V
```

```
V     J=1
V
V     INTEGER P_TYPES(I,J)---------------THE PENETRATOR TYPE USED
      1
V     INTEGER N_PULSE_INIT(I,J)----------THE NUMBER OF ARRIVAL PULSES
F     NPULSEINIT
      10
V
V     FOR K=1,N_PULSE_INIT, SPECIFY THE FOLLOWING:
V
V     K=1
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      0
V
V
V     K=2
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      15
V
V     K=3
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      30
```

```
V
V
V     K=4
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      45
V
V
V     K=5
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      60
V
V
V     K=6
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      75
V
V
V     K=7
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
```

```
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      90
V
V
V     K=8
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      105
V
V
V     K=9
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      120
V
V
V     K=10
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
```

```
        135
V
V
V       FOR I=1,N_RAID_PATHS, SPECIFY THE FOLLOWING:
V
V       I=5
V
V       INTEGER RAID_PATHS(I)---------------THE PATH USED FOR THE RAID
R       PENTIMSEQ
F       RAIDPATHS
        5
V
V       INTEGER N_P_TYPES(I)----------------THE NUMBER OF PENETRATOR TYPES USED
F       NPTYPES
        1
V
R       PATHPEN
V       FOR J=1,N_P_TYPES, SPECIFY THE FOLLOWING:
F       PTYPES
V
V       J=1
V
V       INTEGER P_TYPES(I,J)----------------THE PENETRATOR TYPE USED
        1
V       INTEGER N_PULSE_INIT(I,J)----------THE NUMBER OF ARRIVAL PULSES
F       NPULSEINIT
        10
V
V       FOR K=1,N_PULSE_INIT, SPECIFY THE FOLLOWING:
V
V       K=1
V
F       PULSE
V       REAL        LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
        0.5
V       REAL        TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
        0
V       REAL        TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
        9
V       REAL        TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
        0
V       REAL        T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
        0
V
V
V       K=2
V
F       PULSE
V       REAL        LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
        0.5
V       REAL        TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
        0
```

```
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      15
V
V     K=3
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL     TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      30
V
V
V     K=4
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL     TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      9
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      45
V
V
V     K=5
V
F     PULSE
V     REAL     LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL     TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL     TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL     TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL     T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      60
V
V
```

```
V     K=6
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      75
V
V
V     K=7
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      90
V
V
V     K=8
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      105
V
V
V     K=9
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
```

```
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      120
V
V
V     K=10
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      135
V
V
V     FOR I=1,N_RAID_PATHS, SPECIFY THE FOLLOWING:
V
V     I=6
V
V     INTEGER RAID_PATHS(I)--------------THE PATH USED FOR THE RAID
R     PENTIMSEQ
F     RAIDPATHS
      6
V
V     INTEGER N_P_TYPES(I)---------------THE NUMBER OF PENETRATOR TYPES USED
F     NPTYPES
      1
V
R     PATHPEN
V     FOR J=1,N_P_TYPES, SPECIFY THE FOLLOWING:
F     PTYPES
V
V     J=1
V
V     INTEGER P_TYPES(I,J)---------------THE PENETRATOR TYPE USED
      1
V     INTEGER N_PULSE_INIT(I,J)----------THE NUMBER OF ARRIVAL PULSES
F     NPULSEINIT
      10
V
V     FOR K=1,N_PULSE_INIT, SPECIFY THE FOLLOWING:
V
V     K=1
V
```

```
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      0
V
V
V     K=2
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      15
V
V     K=3
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      30
V
V
V     K=4
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
```

```
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      45
V
V
V     K=5
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      60
V
V
V     K=6
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      75
V
V
V     K=7
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      90
V
V
V     K=8
V
```

```
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      105
V
V
V     K=9
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      120
V
V
V     K=10
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)------ ---THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      135
V
V
V     FOR I=1,N_RAID_PATHS, SPECIFY THE FOLLOWING:
V
V     I=7
V
V     INTEGER RAID_PATHS(I)--------------THE PATH USED FOR THE RAID
R     PENTIMSEQ
F     RAIDPATHS
      7
V
```

```
V     INTEGER N_P_TYPES(I)---------------THE NUMBER OF PENETRATOR TYPES USED
F     NPTYPES
      1
V
R     PATHPEN
V     FOR J=1,N_P_TYPES, SPECIFY THE FOLLOWING:
F     PTYPES
V
V     J=1
V
V     INTEGER P_TYPES(I,J)---------------THE PENETRATOR TYPE USED
      1
V     INTEGER N_PULSE_INIT(I,J)----------THE NUMBER OF ARRIVAL PULSES
F     NPULSEINIT
      10
V
V     FOR K=1,N_PULSE_INIT, SPECIFY THE FOLLOWING:
V
V     K=1
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      0
V
V
V     K=2
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      15
V
V     K=3
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
```

```
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      30
V
V
V     K=4
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      45
V
V
V     K=5
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      60
V
V
V     K=6
V
F     PULSE
V     REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
```

```
       75
V
V
V      K=7
V
F      PULSE
V      REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
       0.5
V      REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
       0
V      REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
       8
V      REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
       0
V      REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
       90
V
V
V      K=8
V
F      PULSE
V      REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
       0.5
V      REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
       0
V      REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
       8
V      REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
       0
V      REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
       105
V
V
V      K=9
V
F      PULSE
V      REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
       0.5
V      REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
       0
V      REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
       8
V      REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
       0
V      REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
       120
V
V
V      K=10
V
F      PULSE
V      REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
```

```
     0.5
V    REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
     0
V    REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
     8
V    REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
     0
V    REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
     135
V
V
V    FOR I=1,N_RAID_PATHS, SPECIFY THE FOLLOWING:
V
V    I=8
V
V    INTEGER RAID_PATHS(I)--------------THE PATH USED FOR THE RAID
R    PENTIMSED
F    RAIDPATHS
     8
V
V    INTEGER N_P_TYPES(I)---------------THE NUMBER OF PENETRATOR TYPES USED
F    NPTYPES
     1
V
R    PATHPEN
V    FOR J=1,N_P_TYPES, SPECIFY THE FOLLOWING:
F    PTYPES
V
V    J=1
V
V    INTEGER P_TYPES(I,J)---------------THE PENETRATOR TYPE USED
     1
V    INTEGER N_PULSE_INIT(I,J)----------THE NUMBER OF ARRIVAL PULSES
F    NPULSEINIT
     10
V
V    FOR K=1,N_PULSE_INIT, SPECIFY THE FOLLOWING:
V
V    K=1
V
F    PULSE
V    REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
     0.5
V    REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
     0
V    REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
     8
V    REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
     0
V    REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
     0
V
```

```
V    K=2
V
F    PULSE
V    REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
     0.5
V    REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
     0
V    REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
     8
V    REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
     0
V    REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
     15
V
V    K=3
V
F    PULSE
V    REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
     0.5
V    REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
     0
V    REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
     8
V    REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
     0
V    REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
     30
V
V
V    K=4
V
F    PULSE
V    REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
     0.5
V    REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
     0
V    REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
     8
V    REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
     0
V    REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
     45
V
V
V    K=5
V
F    PULSE
V    REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
     0.5
V    REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
     0
V    REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
```

```
       8
V      REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
       0
V      REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
       60
V
V
V      K=6
V
F      PULSE
V      REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
       0.5
V      REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
       0
V      REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
       8
V      REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
       0
V      REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
       75
V
V
V      K=7
V
F      PULSE
V      REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
       0.5
V      REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
       0
V      REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
       8
V      REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
       0
V      REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
       90
V
V
V      K=8
V
F      PULSE
V      REAL      LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
       0.5
V      REAL      TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
       0
V      REAL      TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
       8
V      REAL      TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
       0
V      REAL      T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
       105
V
V
```

```
V     K=9
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J.K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      120
V
V
V     K=10
V
F     PULSE
V     REAL       LAMBDA_INIT(I,J,K)---------THE DISTRIBUTION FUNCTION VALUE
      0.5
V     REAL       TAU_R_INIT(I,J,K)----------THE FIRST PULSE PHASE VALUE
      0
V     REAL       TAU_C_INIT(I,J,K)----------THE SECOND PULSE PHASE VALUE
      8
V     REAL       TAU_F_INIT(I,J,K)----------THE LAST PULSE PHASE VALUE
      0
V     REAL       T_PULSES_INIT(I,J,K)-------THE START TIME OF THE PULSE
      135
V
V
```

```
R     PKBLK
V     PEN_ON_UNIT(I,J),J=1,1,I=1,2
F     PEN_ON_UNIT
        0
        0




R     WEPARA
V     INTEGER N_UNITS--------NUMBER OF DISTINCT UNIT TYPES
V       TOTAL NUMBER OF SAMS AND FIGHTERS AVAILABLE
F     N_UNITS
      2
V     INTEGER   N_FI_TYP--------NUMBER OF FI UNIT TYPES
F     NUM_FI_TYPES
      1
V
V     FOR I=1,N_UNITS, SPECIFY THE FOLLOWING
V
V     INTEGER UNIT_TYPES(I)--THE LITERALS DEFINING DISTINCT UNIT TYPES
R     FIWEPARA
F     UNIT_TYPES
      FI1
V     REAL      TOT_FUEL(I)----THE TOTAL FUEL CAPACITY OF THE UNIT TYPE
V       CAPACITY IS IN TERMS OF SECONDS AT CRUISE SPEED
F     TOTFUEL
      14400.0
V     REAL      RAD_RANGE(I)---THE RADAR RANGE OF THE UNIT TYPE (METERS)
F     RADRANGE
      150E3
V     REAL      T_MI_LOAD(I)---------MI LOADING TIME
F     TMILOAD
      0.0
V     REAL      CRUISE_SPEED(I)-------CRUISE SPEED
F     CRUISESPEED
      220.0
V     REAL      DASH_SPEED(I)---------DASH SPEED
F     DASHSPEED
      370.0
V     REAL      CRUISE_TO_DASH(I)----CRUISE TO DASH CONSUMPTION RATIO
F     CRTODASH
      5.0
V     REAL      CRUISE_TO_FIGHT(I)---CRUISE TO FIGHT CONSUMPTION RATIO
F     CRTOFIGHT
      5.0
V     REAL      TURN_RATE(I)---------TURN RATE (RADIANS/SEC)
F     TURNRATE
      0.10
V     REAL      CRUISE_TO_LOITER(I)--CRUISE TO LOITER CONSUMPTION RATIO
F     CRTOLOITER
```

```
      0.43
V     REAL      AUTO_RANGE(I,J)------AUTONOMY RANGES FOR 'TITE' AND 'LOOS'
V                                ------CONTROL MODES (J=1,2)
F     AUTORANGE
      90E3, 150E3
V
V       ********** END OF FIGHTER SECTION  *********************************
V
V       ********** STATS FOR THE SAM WEAPONS SYSTEM ************************
V
V     INTEGER UNIT_TYPES(I)--THE LITERALS DEFINING DISTINCT UNIT TYPES
R     MIWEPARA
F     UNIT_TYPES
      MI1
V     REAL      TOT_FUEL(I)----THE TOTAL FUEL CAPACITY OF THE UNIT TYPE
F     TOTFUEL
      0.0
V     REAL      RAD_RANGE(I)---THE RADAR RANGE OF THE UNIT TYPE
F     RADRANGE
      0.0
V     REAL      T_MI_LOAD(I)---------MI LOADING TIME
F     TMILOAD
      330.0
V     REAL      CRUISE_SPEED(I)------CRUISE SPEED
F     CRUISESPEED
      230.0
V     REAL      DASH_SPEED(I)----------DASH SPEED (MAX FOR MIS)
F     DASHSPEED
      800.0
V     REAL      CRUISE_TO_DASH(I)----CRUISE TO DASH CONSUMPTION RATIO
F     CRTODASH
      0.0
V     REAL      CRUISE_TO_FIGHT(I)---CRUISE TO FIGHT CONSUMPTION RATIO
F     CRTOFIGHT
      0.0
V     REAL      TURN_RATE(I)---------TURN RATE
F     TURNRATE
      0.0
V     REAL      CRUISE_TO_LOITER(I)--CRUISE TO LOITER CONSUMPTION RATIO
F     CRTOLOITER
      0.0
V     REAL      AUTO_RANGE(I,J)------AUTONOMY RANGES FOR 'TITE' AND 'LOOS'
V                                ------CONTROL MODES (J=1,2)
F     AUTORANGE
      0.0,0.0
V
```

```
V     NUM_COMM_INTERVALS -- NUMBER OF INTERVALS OVER
V               WHICH COMMS DATA WAS COLLECTED
F     NUM_COMM_INTERVALS
      2
V     COMM_INTERVAL_TIMES -- START AND END TIMES FOR
V     THESE INTERVALS
F     COMM_INTERVAL_TIMES
      0, 5000
      5000, 10000
V     MEAN_DELAY -- AVERAGE DELAY OVER EACH INTERVAL
F     MEAN_DELAY
      0.00, 0.00
V     NUM_OD_PAIRS -- NUMBER OF LINKS FOR WHICH
V     COMMUNICATIONS WERE OBSERVED IN ADCRM
F     NUM_OD_PAIRS
      0
```

APPENDIX D: DATA/RESULTS

Production Run Data:

ACCURACY	CAPABILITY: .02	.06	.10
.8	# HOSTILES/ FRATRACIDES 69/7	41/13	51/0
.9	37/7	30/2	36/3
.99	12/0	9/0	29/0

ID TIME = 112 SECONDS

ACCURACY	CAPABILITY: .02	.06	.10
.8	# HOSTILES/ FRATRACIDES 64/7	83/5	54/6
.9	39/0	33/3	35/7
.99	12/0	13/0	0/0

ID TIME = 60 SECONDS

ACCURACY	CAPABILITY: .02	.06	.10
.8	# HOSTILES/ FRATRACIDES 60/5	63/9	61/5
.9	61/3	63/0	70/2
.99	44/0	40/0	53/0

ID TIME = 10 SECONDS

Goodness of Fit for Manual ID Time:

Data from Eglin operational test of CIS-ISS in May 1985.

Histogram stats:

Interval			Frequency	Percent	Cum. %
.33	to	.946	26	28.3	28.3
.947	to	1.563	25	27.2	55.4
1.564	to	2.18	17	18.48	73.9
2.181	to	2.80	6	6.52	80.4
2.80	to	3.41	2	2.17	82.6
3.42	to	4.03	4	4.35	87.0
4.04	to	4.65	7	7.61	94.6
4.66	to	5.26	3	3.26	97.9
5.27	to	5.88	0	0.0	97.9
5.89	to	6.50	2	2.13	100.0

Mean = 1.86924 minutes (112 seconds)
Var = 1.94237 minutes
Sample size = 92

Goodness of Fit conducted manually and with AID package. Both indicated that exponential was a "good" fit, but AID showed lognormal to be "best" fit. Thesis scenario used exponential with mean of 112 seconds due to model limitations (e.g. model inputs limited to the same distribution for all service times).

Goodness of Fit for CIS-ISS ID Time:

Data from Eglin operational test of CIS-ISS in May 1985.

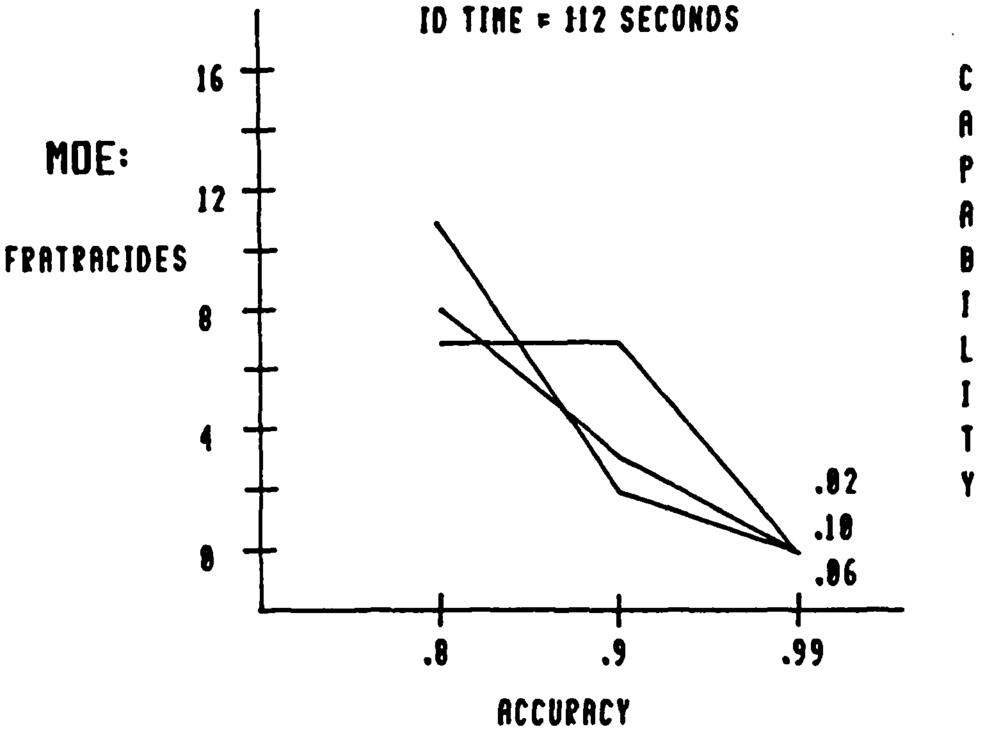
Histogram stats:

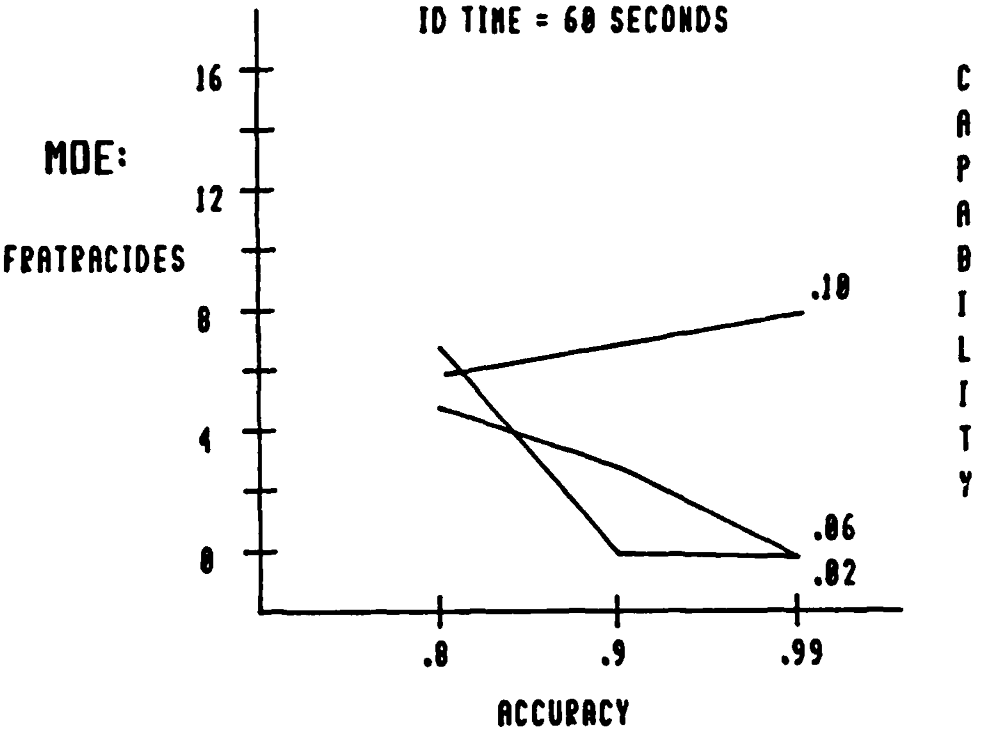
Interval			Frequency	Percent	Cum. %
.1	to	.669	38	52.8	52.8
.67	to	1.24	13	18.1	70.9
1.25	to	1.81	5	6.9	77.8
1.82	to	2.38	2	2.8	80.6
2.39	to	2.95	3	4.2	84.8
2.96	to	3.52	4	5.6	90.4
3.53	to	4.09	1	1.4	91.8
4.10	to	4.66	3	4.2	96.0
4.67	to	5.23	1	1.4	97.4
5.24	to	5.80	2	2.7	100.0

Mean = 1.27806 minutes (78 seconds)
Var = 1.95967 minutes
Sample size = 72

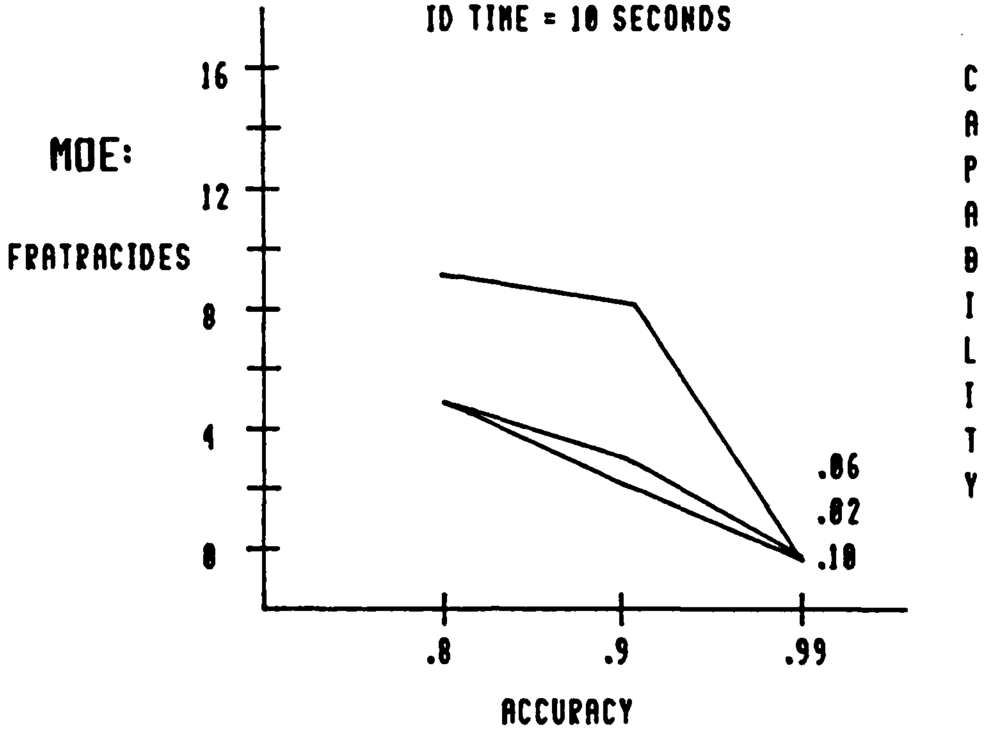
Goodness of Fit conducted manually and with AID package. Both indicated that exponential was a "good fit". Thesis scenario used exponential with mean of 78 seconds.

Interaction Graphs for Fratracide:





Interaction Graphs for Fratracide (continued...):



Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Distribution Unlimited;
Approved for Public Release

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GST/ENS/86M-13

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Air Force Institute
of Technology

6b. OFFICE SYMBOL *(If applicable)*
AFIT-EN

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State and ZIP Code)*
Wright-Patterson AFB
Ohio, 45433

7b. ADDRESS *(City, State and ZIP Code)*

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State and ZIP Code)*

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO

11. TITLE *(Include Security Classification)*
Refer to Block 19

12. PERSONAL AUTHOR(S)
MacFarlane, Maj. Robert Craig and McGuire, Capt. David Michael

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM Aug 85 TO Jun 86

14. DATE OF REPORT *(Yr., Mo., Day)*
86 06 01

15. PAGE COUNT
259

16. SUPPLEMENTARY NOTATION
Prepared in cooperation with the Air Force Human Resources Laboratory.

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Air Defense, Tactical Air Control System,
TACS, CRC, Control and Reporting Center.

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Title: A SIMULATION ANALYSIS OF AN AUTOMATED IDENTIFICATION PROCESSOR FOR THE TACTICAL AIR CONTROL SYSTEM.

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER 3 Sept 86
Dean for Research and Professional Development
Air Force Institute of Technology (AFSC)
Wright-Patterson AFB OH 45433

Control of the airspace over the battlefield is a complex task. The control and reporting center (CRC), as part of the tactical air control system (TACS), plays a vital role in the air defense mission. The purpose of this thesis was to evaluate the utility of the proposed combat identification system - indirect subsystem (CIS-ISS), an automated identification feature, within the CRC. This issue was considered through a comparison of the automated system with the current manual system of identification. The primary measure of comparison was the number

(continued on reverse)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED □ SAME AS RPT. ☒ DTIC USERS □

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
Lt. Col. Joseph W. Coleman, USAF

22b. TELEPHONE NUMBER *(Include Area Code)*
(513) 255-3871

22c. OFFICE SYMBOL
AFHRL/LRA

EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

Block 19. Abstract (continued)...

of hostile aircraft prosecuted during the first "wave" of a conventional attack on U.S. forces by the numerical superior Warsaw Pact forces.

Transient Air Defense Zone (TADZ), a large Fortran and SLAM based simulation model of the Soviet air defense system in use at the Foreign Technology Division, was modified to represent the structure and operating procedures of the TACS. Parametric inputs were made to TADZ based on operational test and performance data for the CIS-ISS and the CRC. The model was then used to provide data for a statistical comparison between the manual and automated identification systems. The results showed that if the CIS-ISS is used in the envisioned centralized location with a "man in the loop," it will backlog under the load of a Central European threat. Distribution of the CIS-ISS from a centralized location and collection of more reliable input data are recommended areas for future effort.

END

10-86

DTIC